UNIVERSIDAD COMPLUTENSE DE MADRID

FACULTAD DE INFORMÁTICA

Departamento de Arquitectura de Computadores y Automática

BIOINSPIRED HEURISTICS FOR THE THERMAL-AWARE FLOORPLANNING OF 3D MPSOCS. HEURÍSTICAS BIOINSPIRADAS PARA EL PROBLEMA DE FLOORPLANNING 3D TÉRMICO DE DISPOSITIVOS MPSOCS

MEMORIA PARA OPTAR AL GRADO DE DOCTOR
PRESENTADA POR

Ignacio Arnaldo Lucas

Bajo la dirección de los doctores

José Ignacio Hidalgo Pérez
José Luis Ayala Rodrigo
José Luis Risco Martín

MADRID, 2013

Bioinspired Heuristics for the Thermal-Aware Floorplanning of 3D MPSoCs

Heurísticas Bioinspiradas para el Problema de Floorplanning 3D Térmico de Dispositivos MPSoCs

Ignacio Arnaldo Lucas

Departamento de Arquitectura de Computadores y Automática

Universidad Complutense de Madrid

Memoria presentada para optar al grado de

Doctor por la Universidad Complutense de Madrid

en el programa de Ingeniería Informática

Directores:

Dr. D. José Ignacio Hidalgo

Dr. D. José Luis Ayala

Dr. D. José Luis Risco Martín

Abril de 2013

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In 2006, Dr Wang, former CEO of Samsung Electronics said "Rapid adoption of 3D integration technology seems to be essential and, thankfully, unavoidable" [71]. Many advances have been presented ever since and commercial products are already available. However, the full potential of 3D integration is yet to be reached.

Although 3D integration was first proposed in the early 1980s, it did not raise a great deal of attention until the mid 2000s, as silicon devices started to approach their limits (see [79]). A lot of resources and effort are being invested nowadays in improving the integration technologies required for the mass scale fabrication of 3D architectures. To explain this trend, it is necessary to summarize the evolution of the semiconductor industry since the mid 1990s. Ever since, not only the performance of the fabricated chips has skyrocketed, but also the released huge computing power has come to be within easy reach with the massive development of mobile devices. However, engineers and designers have come upon many difficulties in this journey. As explained in the following, these issues have been brilliantly addressed in some cases, and perhaps postponed to a near future in others.

Initially, frequency scaling and architectural innovations kept the performance enhancement trend going. However, the increase of clock speed slowed in the mid 2000s due to the fact that pushing frequency above a given limit is not efficient in terms of power consumption. Moreover, the resulting high temperatures negatively affect reliability and lifetime of silicon chips.

Therefore, the industry moved towards architectures with multiple processors working at a lower clock speed to meet the performance requirements imposed by high-demanding applications like multimedia processing. This way, even though frequency stalled or decreased, the number of executed instructions per second continued to rise.

The ever increasing need of performance led to increment the number of cores per die. This increment was made possible by device shrinking together with advances in integration technology and communication architectures, resulting in a higher integration density. Thus, chips from consecutive generations presented a similar area. As a replication strategy has been followed, the number of integrated cores has typically doubled from one generation to the next, resulting in a higher bandwidth demand. Moreover, recent processors have reincorporated Simultaneous MultiThreading techniques, resulting in an increased virtualization and an even stronger need of bandwidth as more bus transactions are required. Figure 1.1 depicts the evolution of the number of integrated processors in a single die, as stated by Stephen Rusu in [129].

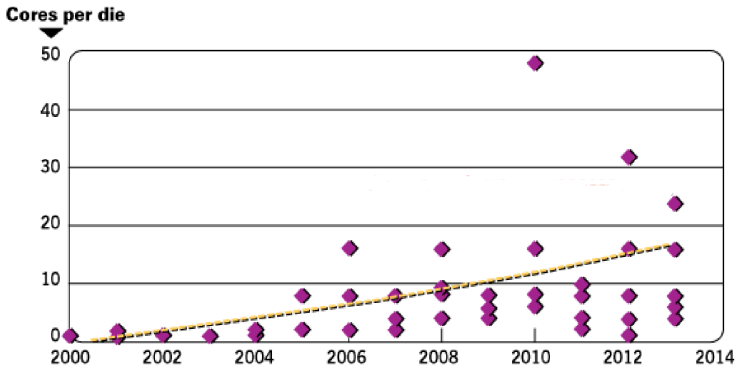


Figure 1.1: Evolution of the number of integrated processors per die as shown in [129]

At the same time, the improvements made to reduce wire delays have not matched the increasing frequency. This issue is becoming more relevant since chip area is expected to increase due to the fact that integration density is reaching its physical limit. In fact, as the transistor size decreases, device leakage current increases while the variability worsens [52]. Since it will not be possible to reduce

transistor size indefinitely, the area needed to integrate more and more cores will significantly increase. As a consequence, the chip area accessible within one cycle will continue to be reduced at every generation. This reduction, in turn, leads to higher communication latencies. Therefore, the increasing and slower transactions will set the limit for the performance improvement provided by the integration of more cores in a single processor [16].

In brief, it can be said that easy performance gains will no longer be possible. The semiconductor industry must therefore make substantial moves to keep up with the demanded performance [91]. Integrating a higher number of processors in a single chip will only be beneficial if an increased bandwidth is provided and the distance between interconnected components is reduced. This is where 3D integration comes into play. In fact, stacking vertically connected ICs leads to an overall wire length reduction and provides high bandwidth connections between layers. Moreover, recent Network-on-Chip (NoC) communication approaches are also compatible with 3D architectures [116]. Broadly speaking, in 3D architectures, total wiring is expected to drop by a factor of $(N_{layers})^{1/2}$, as opposed to the quadratical increase of 2D architectures [44]. For example, a two layer design of a Pentium IV resulted in a 15% performance increment thanks to an improved design of the existing pipelines [102]. Additionally, vertical connections between layers implemented with Through Silicon Vias (TSVs) provide a massive bandwidth. For instance, this higher capacity can be exploited to speedup memory access operations if memories and processors are connected with TSVs.

Therefore, 3D integration is considered a key technology capable of providing the demanded performance enhancement in the coming years. Such technology combined with device scaling was first seen as a way to more than double device density in each generation, resulting in "More than Moore" applications. However, as device scaling becomes harder, 3D chips will more likely be a way to keep on track with the well-known Moore Law [113] ("More Moore") [101]. Additionally, 3D integration has the potential to provide benefits other than performance enhancement. In fact, wire length reduction also results in a significant decrease of both power consumption and timing issues due to clock skew and jitter [102]. Moreover,

heterogeneous integration is allowed as different process technologies can be combined in a single stack. Finally, the modular integration of layers can reduce the overall cost and design effort [52]. In summary, 3D integration provides:

1. Higher performance due to a shorter wire length and a higher bandwith provided by TSVs (vertical distances are shorter).

2. Power reduction thanks to a reduced footprint of clock and power networks, as compared to an equivalent 2D architecture.

3. Heterogeneous integration.

4. Modular integration resulting in a reduced design effort.

As a consequence, 3D integration has risen a great deal of interest in both the industrial and academic worlds. Table 1.1 shows some of the organizations involved in research and development projects related to 3D integration (see [124]).

AMERICA	**ASIA**	**EUROPE**
Albany Nanotech	Amkor	3D-PLUS
Cubic Wafer	ASE	CEA-LETI
Freescale	ASET	EMFT Munich
Ga Tech	Chartered	Fraunhofer IZM
IBM	Elpida	IMEC
Intel	Fujikura	Infineon
Lincoln Labs	Hitachi	NXP
Micro	IME	Sensonor
MIT	ITRI	Siemens
NC State	KAIST	SINTEF
RPI	NEC	STM
RTI	Oki	TU Chemnitz
SANDIA National Labs	Renesas	VTI
Sematech	Samsung	
Stanford	Sanyo	
Texas Instruments	Sharp	
Tezzaron	Sony	
Univ. Arkansas	STATSChipPAC	
Univ. Minn	Tohoku Univ.	
Xilinx	Toshiba	
Ziptronix	TSMC	
	ZyCube	

Table 1.1: 3D activity as shown in [124]

3D integration involves several technological possibilities giving birth to a wide range of end products. Some commercial applications requiring a simple packaging are already commercialized. On the other hand, initiation of mass-market high performance 3D IC products has not yet taken place as technological requirements can not be met with low production costs [124]. Additionally, whether or not adopting 3D technology is a complicated business decision as the associated risk factors are still higher than those corresponding to past changes of process technology. Several fabrication steps have to be improved, such as vias formation, thin-wafer handling, wafer alignment, and bonding [92]. Other major concerns are related to test issues [94] and the lack of Electronic Design Automation (EDA) tools suitable for 3D design [104], [92]. In fact, the industry needs the creation of standards and a tool infrastructure for 3D integration. In particular, 3D place-and-route algorithms, power and timing models, and floorplanning tools are urgently required [56]. Figure 1.2 shows the status of 3D IC research and development as of 2009 [94].

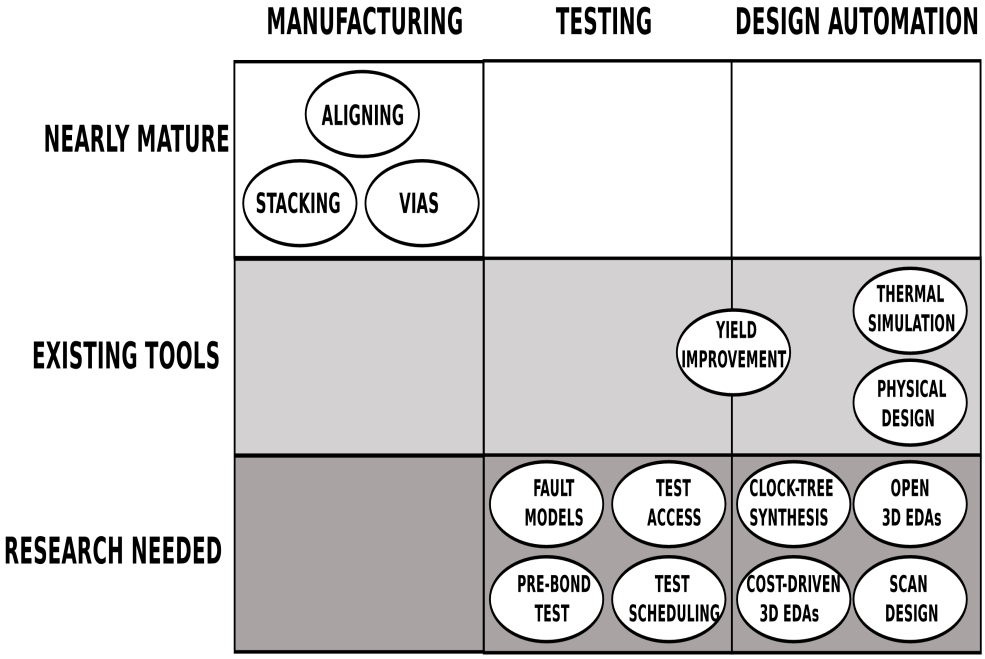


Figure 1.2: Status of 3D IC R&D.

As depicted in the figure, there is a need for design automation tools. Such tools must avoid high temperatures and high thermal gradients as these two factors

negatively impact the performance, reliability and lifetime of silicon chips [44], [90]. These critical issues are magnified in 3D ICs because of two main reasons. First, power density increases with the number of layers as more devices are packed in a smaller area. Second, additional layers are stacked farther from the heat sink, hindering heat dissipation. Therefore, managing thermals is essential for the widespread adoption of 3D integration [102], [52], [104]. However, traditional floorplanners failed to consider thermal restrictions during the optimization process.

Thermal-aware floorplanning is used to reduce the peak temperature of the chip, generally looking for an optimum floorplan that minimizes area, temperature, and wire length (translated into performance). The idea behind this technique is that strictly regular arrangements do not lead to configurations with the required thermal characteristics. In fact, if a hot block is placed beside cooler blocks, lateral spreading of heat takes place. As a result, the temperature of the hot block is reduced [131]. To illustrate these thermal issues, we show in Figure 1.3 the thermal maps of a four-layered 48 cores architecture inspired in the Niagara 2 and Niagara 3 platforms (such architecture will be used as benchmark later in this work). The depicted configuration corresponds to a regular arrangement of the SPARC cores (SPC), Power6 cores (P6), memories (L2) and crossbars (Cross) of the architecture. As a consequence, the SPARC cores are placed above the others producing severe hotspots reaching 411.82K.

Thermal-aware floorplanning techniques have become more and more relevant since device shrinking and frequency scaling have led to higher power densities resulting in higher temperatures. There are a number of proposals targeting the optimization 2D circuits that have been extended to deal with the new 3D platforms. However, they generally fail to provide thermally optimal floorplans in a reduced time. As will be shown in this work, there is a need for new methods better adapted to the three dimensional scenario.

Even though a complete toolchain covering all the steps of the design process has not yet been introduced, there already exist useful tools for 3D architectural exploration. For instance, a number of thermal simulators have been proposed,

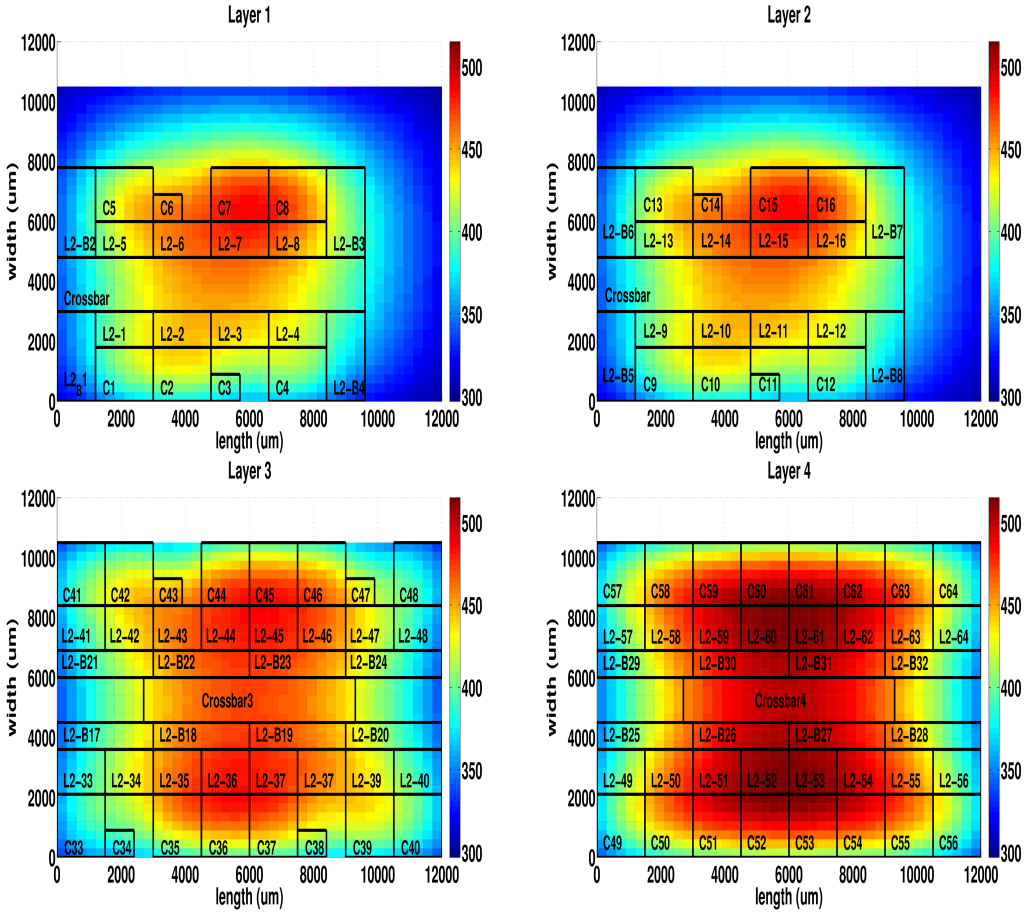


Figure 1.3: Thermal map of the 4 layers of the baseline configuration of the 48 cores platform

generally employed to make sure that candidate floorplans do not exhibit extreme temperatures or elevated thermal gradients [75], [24], [145], [149], and [138]. Another example is the simulation of liquid cooling necessary for a number of 3D platforms, namely for 3D MPSoCs [9], [41], and [130]. As will be shown in this work, the produced cooling effect must be considered in the optimization process to obtain configurations presenting a good tradeoff between performance and thermal behavior.

From a computational point of view, the floorplanning problem is classified as NP-Hard problem (see [17] and [58]). The addition of the third dimension results in an even bigger design space as a greater number of configurations are possible. Moreover, candidate configurations typically present a tradeoff between thermal

behavior and performance, which represent fundamental features of 3D platforms. Thus, appropriate methods are required to explore the solution space in an efficient manner and obtain thermally and performance optimized floorplans.

In this work, Multi-Objective Evolutionary Algorithms (MOEAs) are employed to attack the thermal-aware floorplanning of 3D MPSoCs. These algorithms represent an extension of Evolutionary Algorithms (EAs). In a nutshell, EAs are population-based metaheuristics inspired in Darwin's concept of evolution capable of obtaining good solutions of complex problems in a reduced time. MOEAs, in turn, incorporate the concept of Pareto optimality and are widely recognized as an efficient way of obtaining solutions that present a good tradeoff between two or more conflictive objectives. Therefore, MOEAs represent a suitable tool for the optimization of the targeted architectures, resulting in configurations that simultaneously minimize temperature and maximize performance.

Given the relevance of 3D MPSoCs and the lack of floorplanning tools well adapted for their optimization, we establish the following objectives:

- To make sure that peak temperatures of large 3D MPSoCs remain in acceptable levels while providing an increased performance.
- To study the suitability of existing floorplanning proposals adapted to perform the optimization of the targeted large 3D MPSoCs. In a similar way, it is necessary to compare the validated techniques against new floorplanning representations and algorithms.
- It is necessary to provide manufacturers with a wide range of optimal solutions presenting a good tradeoff between performance and thermal behavior. This way, chip manufacturers can select the most suitable floorplan according to their own criteria.
- To propose parallel thermal-aware floorplanning techniques to speedup the architectural exploration process.
- To compare the suitability of several existing thermal models as guiding cost function in the studied floorplanning context.

- To study whether or not additional cooling techniques such as microfluidic cooling are necessary. If required, consider the resulting effect in the optimization process.

Once the study of the thermal-aware floorplanning of 3D MPSoCs has been motivated and the main objectives of this work have been stated, we describe the structure of this thesis and propose a brief overview of the chapters composing this work.

In Chapter 2, we motivate the exploration of 3D MultiProcessor Systems-on-Chip. To this end, we present the existing 3D integration technologies and study their compatibility with different design possibilities. We also introduce several thermal models used in this work to simulate the thermal behavior of the targeted architectures.

Next, in Chapter 3, we present the most relevant floorplanning proposals and adapt them to optimize three large architectures based on the Niagara platform. The presented comparative study shows that new multi-objective techniques are required to deal with the specific constraints imposed by large 3D MPSoCs.

The Multi-Objective Evolutionary Algorithms adopted in this work to attack the thermal-aware floorplanning problem are explained in Chapter 4. To introduce these methods, a brief introduction to multi-objective optimization is proposed. We also present Evolutionary Algorithms and the main parallelization strategies typically followed to accelerate these techniques.

Chapter 5 gathers several contributions of this work. First, a parallel implementation of an existing multi-objective evolutionary floorplanner is presented in order to speedup the optimization. We also introduce additional knowledge to the problem by means of a power profiling step previous to the thermal-aware floorplanning process to better guide the search. Finally, the performance and thermal optimization achieved integrating several thermal models in the floorplanning process is compared.

A new representation, namely Direct Mapping, is proposed in Chapter 6. Such representation, in combination with a MOEA designed to perform the optimization of 3D MPSoCs, leads to outperform previous floorplanning proposals. Additionally, the new floorplanner is guided by thermal simulations that consider the inclusion of microfluidic channels in intermediate layers of the 3D stack.

Even though the previous chapters include partial conclusions, the most relevant ones are summarized in Chapter 7. Future improvements and new research possibilities are also discussed in this chapter. Finally, we provide the references of the publications resulting from the presented research as well as the projects and grants with which this work has been funded.

Finally, the motivations, proposals and most relevant results of this thesis are summarized and translated to Spanish in Appendix A.

Chapter 2

3D Integration

3D Integration is widely recognized as a viable solution to provide increased performance and chip footprint reduction in the absence of scaling. The performance improvement is allowed by a decreased total wiring length, and thus reduced interconnect delay times, an increased number of vertical interconnects, and heterogeneous integration.

3D Technology can be broadly defined as any technology that stacks vertically interconnected semiconductor elements. Under this definition, this technology comprises a wide range of applications that differ from each other, mainly, in the required interconnect density. In fact, some systems require only a few Through-Silicon Vias (TSVs) while others could need extremely high densities (in the range of 100 $pins/cm^2$) [159]. A great deal of applications exist requiring the full range of possible interconnect densities.

Thus, the term 3D Integration has been used in the last ten years to report a wide range of applications and technologies that present tradeoffs between performance, fabrication cost, power consumption and reuse possibilities among others. Some applications such as 3D Packages are already commercially available while other approaches have only been demonstrated with prototypes. For instance, [115] reports the fabrication of a 3D Image sensor with vias. Another example is the report by M. Koyanagi *et. al.* [85], in which the authors present the successful fabrication of four different prototypes using 3D TSV technology: a microprocessor

test chip, a memory test chip, an image sensor chip, and an artificial retina chip.

When it comes to the 3D Integration processes conceived to burst the performance of existing high-performance systems, it can be said that the allowing technology is not yet ready for mass production. However, some of the involved steps are already mature, and the first commercially viable products are expected in the short term future. It has been showed that such architectures have the potential to provide huge benefits in the context of high performance systems. For example, Jacob *et. al.* [78] stated that 3D can break the memory wall. A similar work [99] studied the performance of a single-core processor in different 3D scenarios.

A lot of effort and resources have been dedicated to develop 3D Integration technologies in the last years. As a consequence, the technological constraints imposed by the different 3D processes such as via pitch and via density are evolving quickly, making it difficult to develop universal tools for the design of 3D circuits. As a matter of fact, there is currently a clear gap between technological constraints and design approaches as the latter are based on assumptions rather than on specific technological processes. It is therefore difficult to make the link between the different integration approaches and their potential applications.

This chapter attempts to motivate the architectural and design choices made in this work, namely the exploration of 3D MultiProcessor Systems-on-Chip. The first two sections present the state-of-the-art of the technologies allowing the fabrication of 3D platforms and their compatibility with the emerging design possibilities. The last section introduces the thermal models used later in this work to guide the optimization of 3D chips.

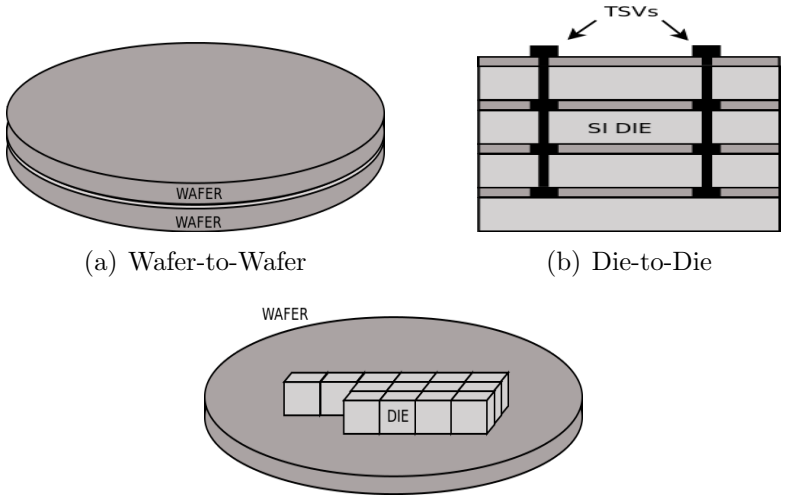
2.1 3D Integration Approaches

In this section we briefly present the different methods and technologies involved in the fabrication of 3D chips. Note that, as the allowing technology is evolving quickly, so is the employed terminology leading to certain confusions. In this section, we classify, label and describe the different 3D Integration alternatives to avoid the

misunderstandings caused by the changing terminology in the remaining of this work.

2.1.1 Stacking Alternatives

Nowadays, several processes leading to three dimensional architectures or circuits have been proposed and demonstrated. In a nutshell, entire wafers can be stacked (bonded) in top of other wafers, and then diced to obtain separate 3D chips. Individual dies can also be stacked directly and there are other hybrid combinations such as Die-to-Wafer stacking involving partial and complete wafers. These different possibilities are illustrated in Figure 2.1



(a) Wafer-to-Wafer (b) Die-to-Die

(c) Die-to-Wafer

Figure 2.1: Schematic representation of major 3D stacking approaches

Wafer-to-Wafer In the Wafer-to-Wafer approach (see Figure 2.1(a)), components are fabricated with two or more wafers. These wafers must be aligned, bonded and finally diced into several 3D chips. Full-wafer bonding is harder to align but provides the best manufacturing throughput [102]. However, a reduced yield is obtained as the stacked dies composing a 3D chip can not be tested separately. As a consequence, the entire chip fails if one of the stacked dies is defective.

Die-to-Die Figure 2.1(b) shows a schematic representation of the Die-to-Die approach. In this case, components are built on separate dies. These dies are typically thinned, aligned and bonded. The different dies forming the stack can be tested separately, leading to an increased yield.

Die-to-Wafer The Die-to-Wafer, as depicted in Figure 2.1(c), is a hybrid approach in which electronic components are built on two wafers. One of the wafers is then diced, aligned and bonded in specific sites of the second wafer. Additional dies can be stacked before dicing.

Even though each of these stacking alternatives presents pros and cons from a cost or yield perspective, the current trend in industry seems to be in favor of die-to-die stacking [94].

Besides the stacking possibilities mentioned above, vertical connections are generally used to classify the different 3D Integration schemes. Namely, most of the state-of-the-art reviews clearly distinguish between the architectures with and without Through-Silicon Vias [92] [124]. In fact, according to [124], the spectrum of 3D Integration technologies can be classified in the following main categories:

1. Stacking of packages or embedded dies without TSVs, also called 3D packaging
2. 3D TSV technology divided in:
    - 3D Integrated Circuits (3D-IC) with high TSV densities
    - 3D ICs with low TSV densities, also called 3D System-on-Chip

We now present an overview of the characteristics of these different 3D Integration approaches.

2.1.2 Packaging-Based Chip Stack

3D packaging reduces the chip footprint by stacking separate chips in a single package. However, this packaging does not integrate the different chips into a single circuit. In fact, off-chip signaling is used for communication, just as if the stacked

chips were placed in separate packages on a 2D circuit board. Figure 2.2 shows a schematic representation of a 3D package with off-chip signaling. A wire bonding strategy is used in this case. Note that there are other alternatives not illustrated here such as flip-chip bonding, however they both present the same main features.

Figure 2.2: Schematic representation of a 3D stack with off-chip signaling

This approach rose a great deal of interest in the late 90s, when 3D packaging was seen as a way to provide wire and area reduction. The latter was especially desirable in the context of an increasing demand for low power consumption, low weight and compact portable devices [2]. It also started to be clear that wiring was to become a limiting factor as it was unable to keep up with the performance enhancement provided by technology scaling [86]. The improvement of the allowing technologies made this research trend continue and gain interest in the following years [148], [96].

This approach was demonstrated for the fabrication of memories and real-time image processing chips [71]. However, several physical design and technological issues common to most of the 3D fabrication processes were detected [96]. These technological issues will be discussed later in this chapter.

Nowadays, 3D packaging is already mature and is at the core of commercially available products since 2006, being widely used in cell phones [104]. However, considering on-chip vertical connections between layers by means of Through-Silicon Vias allows for further wire reduction and for higher integration density [82].

3D Packages present a poor cost-effectiveness ratio due to a number of redundancy/repair issues. In this respect, 3D Integrated Circuits are able to overcome these limitations through easy implementations of redundancies and repairs. In fact, silicon technology based on 3D Integration has been drawing much attention because it regarded as the only practical solution to keep the performance enhancement trend going [71].

2.1.3 3D Integrated Circuits

The term 3D IC typically designates a 3D chip with on-chip signaling. This means that the different layers of the stacked architecture communicate vertically by means of TSVs. In such case, the selection of the integration technology is crucial as it impacts the achievable die-to-die (d2d) via pitch, which in turn impacts the vertical interconnect capability. In fact, both the vertical pitch and bandwidth requirements play a key role to determine the appropriate integration approach for a given application. Other decision factors are silicon efficiency, complexity, thermal management, speed, and power consumption [2], [159].

Most authors distinguish between 3D ICs with very high or low TSVs densities. The first approach is generally employed for transistor-level 3D designs as it requires fine-pitch vias, and TSVs density and via pitch are inversely correlated. In this case, the vias are used as wire to connect different transistors of a CMOS circuit. The second, usually called 3D System-on-Chip (3D SoC), is employed to design 3D circuits at a coarser granularity level. For instance, a 2D processor can be repartitioned into several layers connected with TSVs or 3D MPSoCs can be designed.

Transistor-level Build-ups The fabrication of Transistor-level Build-ups is allowed by several technological processes such as *Transistors Formed Inside On-Chip Interconnect*, *Transistors Formed on Single-Crystal Silicon Films*, or *Transistors Formed on Polysilicon Films*. Nowadays, the applications resulting of these techniques are respectively signal amplifiers for optical interconnects, low-performance nonvolatile memory, and very high-density SRAMs and NANDs [104]. While the three techniques seem promising for the fabrication of future commercialized products, only the latter will provide a significant performance enhancement. This ap-

proach requires an important effort as 3D designs have to be made from scratch. The fact that memories present a regular structure makes them good candidates for this technological process. As will be discussed later in this chapter, it is not expected that more complex and irregular circuits will benefit from this 3D Integration paradigm in the short term.

As an example, we illustrate the approach to manufacture transistors on polysilicon films layer by layer with tungsten interlayer vias in Figure 2.3. In this case, a small piece of silicon film is deposited followed by either laser heating or rapid thermal anneal to recrystallize the silicon. The transistors are then formed by Back-End-Of-Line compatible processing.

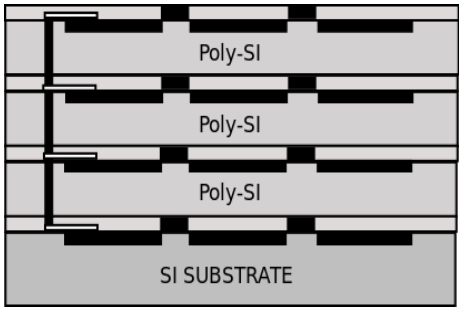


Figure 2.3: Transistors Formed on Polysilicon Films

3D ICs with low TSVs densities In this case, devices are stacked and interconnected at a global level with much lower TSV densities. Note that 3D ICs with low TSVs densities are also referred to as 3D Systems-on-Chip. There are several demonstrations of the fabrication of 3D chips following this approach. For instance, in 2006, J.A. Burns reports the fabrication of three 3-tier digital an analog small scale circuits: a 3D ring oscillator, a 1024×1024 visible imager, and a 64×64 Geiger mode laser radar chip [19]. This approach has been predicted to be a key technology to deal with the so-called "wiring crisis". In fact, further transistor integration densities can be reached, allowing to keep up with the Moore Law ("More Moore"). Moreover, it is compatible with heterogeneous integration technologies [124] and could result in "More than Moore" applications if combined with further transistor scaling. Both academic and industrial organizations are developing and evaluating

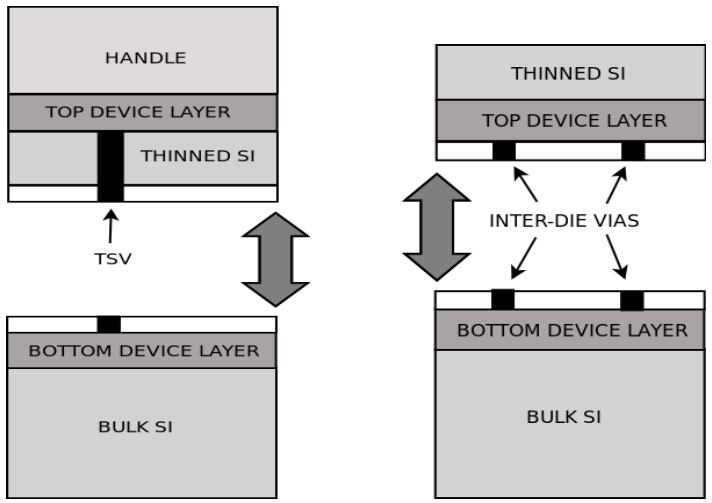
a variety of such 3D Technologies [104]. However, as of today, performance needs can not be met by current TSV technologies with sufficient low production costs [124].

2.2 Difficulties and Challenges

3D Integration targets a wide range of applications presenting different power and performance requirements. Therefore, different process flows are needed in each case. However, some technological processes are common and critical in all the 3D Integration approaches. These crucial steps are briefly described in this section.

2.2.1 Strata Orientation

The orientation of the die in the 3D stack, namely Face-to-Back or Face-to-Face, has important implications for design. The choice impacts the distance between the transistors in different layers, and more importantly, it determines the fabrication process of the 3D chip. Figure 2.4 shows two two-layered 3D chips formed with the Face-to-Face (Figure 2.4(b)) and Face-to-Back approaches (Figure 2.4(a)). Note that, in a 3D chip, multiple strata orientations can be combined, however the implications found in a two-die stack can be used to illustrate the most relevant issues.



(a) Face-to-Back (b) Face-to-Face

Figure 2.4: 3D fabrication methods: Face-to-Face and Face-to-Back.

Face-to-Back

The "Face-to-Back" method bonds the front side of the bottom die with the back side of the top die. Similar approaches were developed for 3D packages at IBM and for the fabrication of CMOS and MEMS applications [159] [144].

Figure 2.4(a) depicts a two-layer 3D chip following the Face-to-Back approach. The final height of the chip basically depends on the thickness of the top wafer, usually thinned before the bonding process. This technique requires a handle wafer that can introduce distortions that make alignment difficult. The typical thickness of the top wafer is on the order of 25-50μm, which limits the via pitch to values on the order of 10-20μm. It is expected that technological advances will help reducing these numbers in the coming years [159]. In this work, the Face-to-Back strategy is adopted to design large 3D MPSoCs as it allows for homogeneous vertical connections when more than two layers are stacked.

Face-to-Face

Figure 2.4(b) shows the "Face-to-Face" approach which joins the front side (face) of two wafers. With this technique, it is possible to achieve much higher interconnect densities than with the Face-to-Back assembly. The via pitch is limited by the alignment tolerance of the bonding process. As bonding systems reach tolerances of 1-2 μm, it is predicted that interconnect pitches of 10 μm or smaller will be achieved with this approach. However, this connectivity between layers would only be feasible for two-layer stacks. Otherwise, the via pitch will still be limited to values on the order of 10-20μm [159].

2.2.2 Wafer Thinning

Wafer thinning allows the distance between the different layers to be reduced, allowing a high density of vertical interconnects. The great challenge of the thinning process is the uniformity requirement, typically $< 1-2\mu m$, with no natural etch stop in the case of bulk Si. There are demonstrations of robust processes in which a great uniformity is achieved but the thickness of the stacked dies have to be $> 20\mu m$. To ease this technological difficulty, several approaches consider the use of an etch-stop

such as an ion implanted layer, a graded SiGe layer, or a buried oxide layer [159].

2.2.3 Alignment

As mentioned above, the alignment tolerances have an impact on the achievable via pitch, and thus, on the density of vertical interconnects. The alignment tolerance depends on the stacking process. Die-to-Die stacking allows for tolerances in the range of $1 - 20\mu m$ depending on the speed of the assembly process. As of 2012, Wafer-to-Wafer systems can achieve an alignment accuracy of $1\mu m$ or smaller

2.2.4 Bonding

The bonding process of two dies presents several mechanical, electrical, and thermal constraints and is a key issue being addressed. To be able to determine the right bonding alternative, the alignment, pre-bonding, and actual bonding must represent different steps of the process flow. In [159], Albert M. Young and Steven J. Koester report three different technological processes to perform the bonding step developed at IBM.

The first one is the Copper-to-Copper Compression Bonding in which a thermo-compression bond is used to attach two wafers. This bond can provide electrical connection between the layers but requires elevated temperatures greater than allowed by CMOS BEOL processes (BEOL metalization typically needs the bonding process to be performed at temperatures lower than 400°C).

The second is the Hybrid Cu/Adhesive Bonding which is a variation of the previous approach in which the adhesive provides an improved bond strength. Moreover, it increases mechanical integrity and presents a higher thermal stability. However, this method is still in a development phase.

The third technique reported by IBM is called Oxide-Fusion Bonding. This approach also requires low temperature processes and extreme planarization. On the other hand, this approach is expected to provide very high-density interconnects in a near future [159] [104].

2.2.5 TSVs

TSVs are at the core of 3D Integration as they are fundamental to really take advantage of the third dimension from a performance perspective. TSVs have been developed in the production environment in companies such as IBM, Toshiba, and ST Microelectronics [159]. Barely speaking, TSVs are vertical holes created in silicon using an etching and filling process. There is a number of reports on the successful fabrication of TSVs see [104] and therein and [85].

Via etching: The etching refers to the creation of the via. Several alternatives such as laser drilling or Deep Reactive Ion Etching (DRIE) have been demonstrated [104].

Via filling: The filling corresponds to the deposition of a highly conductive material in the walls of the constructed via. Copper (Cu) and Tungsten (W) are usually employed to this end. Note that the latter allows for better aspect ratios [159].

Dimensions: As mentioned in Section 2.2.1, TSVs dimensions depend on silicon thickness (specially Cu vias), and other process considerations such as wafer alignment and thinning and represents a key feature for 3D circuits designers. In fact, the floorplanning of the different components of a system has to respect the via placement. Moreover, the placement of such TSVs impacts the overall wire length, and thus, the performance, of the 3D chip.

Moment in the process flow: According to Albert M. Young *et. al.* [159], the timing of creation of the TSVs can be classified in the following categories: Pre-backend frontside via, Post-backend frontside via, and Backside via etch. These approaches impose different design rules and restrictions such as low aspect ratios in the case of the Backside Via procedure or the incompatibility with high complexity designs presented by the Post-backend Frontside Via process. For further details on this topic, the reader is referred to [159].

2.2.6 Test and Yield

From an industrial point of view, yield, test, and throughput drive the cost of 3D Integration [104]. 3D Integration could ideally improve yield since it allows heterogeneous integration. This means that different components of a system can be fabricated in different wafers with different fabrication procedures optimized for the involved technology [94].

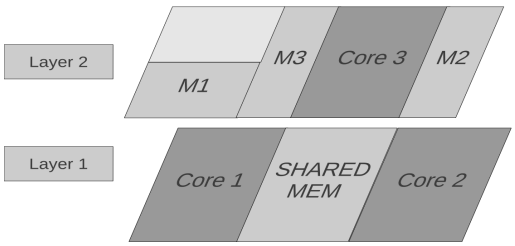
However, several test challenges have been identified. Broadly speaking, to increase yield and ease the testing of the final product, each layer should be independently testable. This presents serious difficulties because testing must be accomplished with very few probe contacts and few wafer touchdowns and also because layers might not present complete functionality [52]. In fact, yield and test issues are sometimes regarded as the biggest challenge of 3D Integration [94]. The adaptation of well known techniques such as redundancy, fault tolerance, Error-Correcting Codes (ECC), Design-For-Testability (DFT), or Built-In Self-Test (BIST) to 3D chips will be fundamental for the commercial success 3D Integration.

2.3 3D Design Possibilities

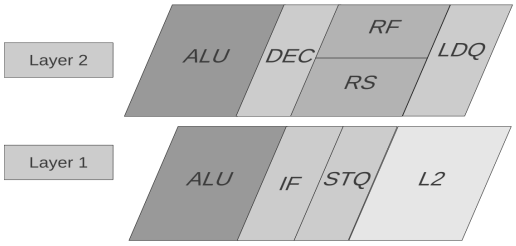
The design of 3D ICs can be considered at different abstraction levels corresponding to different design possibilities. The referred 3D approaches range from coarse to fine grained, depending on the granularity of the unit to be stacked. Figure 2.5 shows the different possibilities of 3D Integration from higher to lower level of abstraction. According to the classification proposed by G. Loh *et. al.* in [102], four different granularities can be considered, namely *Entire cores and memories*, *Functional unit blocks* (FUBs), *Logic gates* (also known as FUB splitting), and *Transistors*. The benefits and redesign effort of the different approaches is summarized below.

2.3.1 Entire Cores and Memories

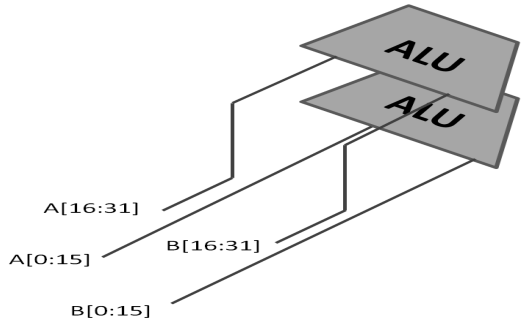
The coarser approach consists in considering entire cores and memories as showed in Figure 2.5(a) where a two-layer 3D MPSoC is depicted. In this case, the designer manages macroscopic two dimensional blocks that must be placed on a single layer



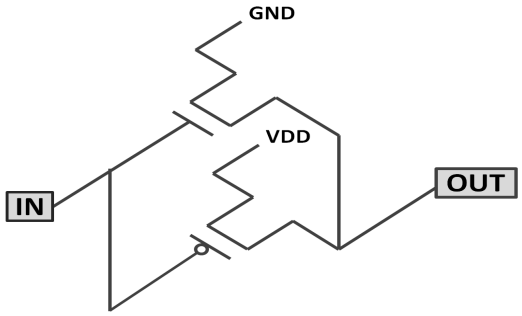
(a) Entire cores and memories



(b) Functional unit blocks



(c) Logic gates: ALU on ALU



(d) Transistors: CMOS inverter

Figure 2.5: Different granularity approaches for 3D Integration.

of the stack. Thus, 2D processors, memories, and Intellectual Property (IP) blocks can be reused, resulting in a low redesign effort. Note that, even if the power and performance of the different components of a system remain unchanged, it allows to reduce the footprint of clock, power networks and global wiring of the whole chip, resulting in a better overall performance. Therefore, the performance of large 3D MPSoCs can be increased as compared to a 2D equivalent architecture. Other examples resulting of this approach are Memory-on-Core architectures.

2.3.2 Functional Unit Blocks

The design of 3D circuits at a Functional Unit level reduces the latency and power of global routes, providing simultaneous performance improvement with power reduction [102]. However, paths must be re-floorplanned and retimed. There is also a lack of 3D place and route tools, making this approach harder. This approach allows to take advantage of existing designs as 2D functional blocks can be reused. For example, Figure 2.5(b) shows a repartition of a 2D processor in two layers.

2.3.3 Logic Gates

The partition and floorplanning of logic circuits in three dimensions at a logic gate level leads to further area reduction due to compact layout and resizing opportunities. On the other hand, new 3D designs, methodologies and layout tools are a must because this approach requires a high interconnect density between stacked layers that has to be taken into account. Figure 2.5(c) shows an ALU bit-splitting resulting in a shorter wire length (as long as vias are properly placed for inter-layer communication).

2.3.4 Transistor

The design at a transistor level presents an extreme cost as it offers almost no reuse possibilities. However, area, power, and latency could theoretically be obtained in the case of large and complex gates. Figure 2.5(d) shows a CMOS inverter presenting a reduced area as compared with the 2D design. However, it is not likely that technology will provide in the near future the extreme interconnect density necessary for 3D CMOS complex designs [102]. Moreover, new 3D methodologies, and

layout tools would also be required to operate at this granularity level.

These approaches clearly present a tradeoff between the potential benefits (performance gain and area and power reduction) and reuse possibilities summarized in Table 2.1 (see [102]).

Stacking unit	Benefits	Redesign effort	2D tools reuse
Cores and Memories	Medium	Low	High
Functional units	Medium	Medium	Medium
Logic gates	High	High	Low
Transistors	High	Extreme	Low

Table 2.1: Benefit and redesign effort of the different 3D design approaches

2.4 3D Integration vs. 3D Design

To select the most suitable design approach for a given application, the designer needs to consider the reuse effort and the available technology processes. From a technological point of view, the choice is mainly driven by the size and the pitch of the TSVs. In fact, fine granularity designs require a small via pitch and a high vertical interconnect density. If the d2d via pitch is large as compared to the unit to be stacked, the designer must consider a coarser partition strategy [102]. Therefore, the different design approaches are only compatible with some of the technologies described in Section 2.1. We summarize in Table 2.2 the compatibility between design approaches and integration technologies.

Stacking unit	Cores and Mems	FUBs	Logic gates	Transistors
3D Packaging	Medium	Low	Not Feas.	Not Feas.
3D IC high TSV	High	High	High	High
3D IC low TSV	High	Medium	Not Feas.	Not Feas.

Table 2.2: Compatibility of 3D design approaches and manufacturing technologies

Only the Entire Cores and Memories approach is compatible with 3D Packaging. In that case, the performance enhancement, if any, will not be significant as the connections between layers have to be done in an off-chip manner.

3D ICs with high vertical interconnect density are compatible with the four design approaches as any range of inter-layer connectivity can theoretically be provided.

3D ICs with low TSVs densities are only compatible with the two coarser approaches, namely Entire Cores and Memories and FUBs. Moreover, following this integration approach, only the first would fully benefit from a 3D organization as few TSVs are required to improve the overall wire length of Systems-on-Chip.

To motivate the study of 3D MPSoCs performed in this work, it is important to complete Table 2.1 with the information about the maturity state of the required technology for the different design approaches. To this end, Table 2.3 presents the tradeoff between potential benefits, reuse possibilities, and the state of the allowing technology [102].

Stacking unit	Technology ready	Benefits	Redesign effort	2D reuse
Cores and Mems	Short	Medium	Low	High
Functional units	Short	Medium	Medium	Medium
Logic gates	Long	High	High	Low
Transistors	Long	High	Extreme	Low

Table 2.3: Benefit, redesign effort and state of the technology necessary for the different 3D design approaches

3D ICs with a high level of integration and designed at a Logic Gate or Transistor level are only expected in a distant future. The benefits obtained with this approaches could be huge as the potential of 3D Integration would be fully exploited. On the other hand, 3D designs at a coarser granularity level are expected in the short term [124] as no technology show-stoppers are expected [104]. However, 3D Integration is still in a development phase and techniques such as aligning, thinning

and bonding are continuously being improved and are not yet ready for mass-scale production.

The allowing technologies are not mature yet because, even if there are several demonstrations of fabrication of 3D ICs, the end products are not yet commercially viable. However, a lot of effort from the industrial and academic worlds is being invested in research and development of these architectures as 3D ICs are accepted as a viable way to provide the demanded performance. Not only the fabrication costs are still too high but also there are challenges that remain unresolved. While some of these challenges are technological such as test and yield difficulties, others are related with design issues. In fact, there is a need of 3D Electronic Design Automation (EDA) tools. The lack of such tools might be motivated by the absence of technological standards. This means that designers are not aware of the specific technological constraints imposed by a given technology (via pitch and density, interlayer distance etc.). It is therefore, extremely difficult to come up with efficient and universal tools with the current wide range of possible technological scenarios.

However, it is clear that managing thermals is a key challenge for 3D Integration. In fact, significant thermal issues appear mainly because of the two following reasons [102]:

1. The distance between active devices and the heat sink increases with the number of layers.

2. The chip power density increases linearly with the number of layers

As a result, 3D Integration together with the combined scaling of technology and operating frequency might lead to a dramatical increase in maximum chip temperatures. These uneven peak temperatures cause gradient temperatures on the chip surface that negatively affect the reliability and lifetime of integrated circuits. As a result, thermal restrictions have become a major constraint in the design of current 3D ICs. The temperature of a 3D chip depends on the power dissipated by the elements composing the stacked ICs and on the process of lateral and vertical spreading of heat. Therefore, both factors must be taken into account to design architectures with an appropriate thermal behavior. Even though the relevance of the former

varies through time, it is clear that hot blocks need to be placed strategically to enforce heat dissipation. It is therefore **necessary to introduce thermal models in the optimization loop to deal with the imposed thermal restrictions**.

2.5 Thermal Modeling of 3D ICs

In the first part of this section, we propose a brief description of the thermodynamic laws involved in the process of heat diffusion and heat dissipation of 3D ICs. The second part shows the different thermal models used later in this work to guide the thermal optimization process.

2.5.1 Heat Diffusion and Heat Dissipation in 3D ICs

To introduce and motivate the thermal models presented later, we report the equation governing heat diffusion via thermal conduction in a 3D stack [18]:

$$\rho c \frac{\partial T\left(\vec{r},t\right)}{\partial t} = \nabla \times \left[k(\vec{r})\nabla T(\vec{r},t)\right] + p(\vec{r},t) \tag{2.1}$$

where ρ is the material density, c is the mass heat capacity, $T\left(\vec{r},t\right)$, and $k\left(\vec{r}\right)$ are the temperature and thermal conductivity of the material at position $\vec{r}$ and time t, and $p(\vec{r},t)$ is the power density of the heat source. The equation is subject to the following boundary condition:

$$k(\vec{r},t)\frac{\partial T(\vec{r},t)}{\partial n_i} + h_i T(\vec{r},t) = f_i(\vec{r},t). \tag{2.2}$$

With respect to (2.2), n_i is the outward direction normal to the boundary surface i, h_i is the heat transfer coefficient and f_i is an arbitrary function at the surface i.

Discretization Method

A seven point finite difference discretization method can be applied to Equation (2.1) to perform numerical thermal analysis over the three dimensional volume of the IC. To this end, the 3D stack is decomposed into numerous rectangular parallelepipeds. Note that these blocks can be of non-uniform sizes and shapes if necessary. Heat

diffusion takes place between these blocks, each one characterized by its power dissipation, initial temperature, thermal capacitance and thermal resistance to adjacent elements. We now show the discretized equation at an inner point of the so-constructed grid:

$$\begin{aligned} C_v\ V\ \frac{dT}{\Delta t} = & \quad A_x\ (T_{i-1,j,l} - 2\ T_{i,j,l} + T_{i+1,j,l}) \\ & + \quad A_y\ (T_{i,j-1,l} - 2\ T_{i,j,l} + T_{i,j+1,l}) \\ & + \quad A_z\ (T_{i,j,l-1} - 2\ T_{i,j,l} + T_{i,j,l+1}) \\ & + \quad V\ p_{i,j,l} \end{aligned} \tag{2.3}$$

where $C_v = \rho c$ is the volumetric specific heat of the material, i, j and l are discrete offsets along the x, y and z axes, Δt is the discretization step in time t, Δx, Δy and Δz are discretization steps along the x, y and z axes and $V = \Delta x \Delta y \Delta z$. Finally, A_x, A_y and A_z are the thermal conductivities between adjacent elements, defined as follows:

$$A_x = k\frac{\Delta y \Delta z}{\Delta x},\ A_y = k\frac{\Delta x \Delta z}{\Delta y}\ \text{ and }\ A_z = k\frac{\Delta x \Delta y}{\Delta z}. \tag{2.4}$$

Considering a 3D stack composed of N discretized elements, Equation (2.3) can be summarized as follows:

$$\mathbf{C}\frac{dT(t)}{dt} + \mathbf{A}T(t) = Pu(t) \tag{2.5}$$

where the thermal capacitance matrix $\mathbf{C}$ is an $N \times N$ diagonal matrix, the thermal conductivity matrix $\mathbf{A}$ is a $N \times N$ sparse matrix, $T(t)$ and P are $N \times 1$ temperature and power vectors and $u(t)$ is the unit step function.

Steady-state Thermal Analysis

The peak temperatures reached by the different configurations of a floorplan are generally used as a metric to compare their thermal behavior. We assume that these maximum temperatures are registered in the steady-state as the studied systems typically start at ambient temperature. To perform the referred steady-state thermal

analysis, the left term in Equation (2.5) can be dropped, resulting:

$$\mathbf{A}T = P \Leftrightarrow T = \mathbf{A}^{-1}P \tag{2.6}$$

Thus, given the thermal conductivity matrix $\mathbf{A}$ and power vector P, the steady-state thermal analysis is reduced to the computation of the inverse matrix of $\mathbf{A}$.

During the execution of the floorplanning algorithm, blocks are continuously moved searching for a configuration with a better thermal behavior. These operations have as unique effect a change in the distribution of the non-zero coefficients (i.e. power values) in the input vector P. Therefore, the computation of $\mathbf{A}^{-1}$ can be run only once, saved, and reused to compute the thermal profile T every time that a new configuration of blocks must be tested. This is possible as long as the parameters involved in the boundary conditions of the system as expressed in Equation (2.2) do not change during the optimization process. The referred parameters are the volume of the 3D stack, the properties of the material, the heat transfer rate between the IC and the environment, and the discretization applied to compute the numerical solution.

In this thesis, we do not take into account the impact of the leakage current in the final temperature of the different blocks. Such current would increment the power consumption of the different components by a constant factor, thus modifying the values of power vector P of Equation (2.6). Note that none of the methods presented in this work must be modified to consider the effect of leakage power in the final temperature of the studied architectures.

2.5.2 Existing Thermal Models for 3D ICs

In [31], it is made clear that the existing thermal models present a tradeoff between runtime and quality. In fact, the different thermal models are classified in three different categories:

1. Numerical computing analyses (such as Finite Element Method and Finite Difference Method)

2. Compact Resistive Network

3. Simplified closed-formula

Note that the first two methods are accurate and time consuming while the latter is fast but lacks of precision. In the following, we detail extensively the approaches used in this work to guide the thermal optimization of 3D architectures. Special attention is given to the way the different models deal with the aforementioned tradeoff between the accuracy of the results, the complexity of the model and the consequent runtime.

Compact and Transient Thermal Model

In [138] the authors present a compact and transient thermal model to run fast but accurate thermal simulations of 2D or 3D ICs. This model exploits the similarities between heat transfer laws in solid materials and electric current. As shown in Figure 2.6, the volume of a 2D IC is divided into cuboids or *thermal cells* representing a node in an electric circuit with six resistances and one capacitor. The capacitor represents self heat storage while the resistances connect each cell to its neighbors, modeling the heat flowing within the volume. Heat diffusion to the surrounding environment is also considered and modeled connecting the resistances in the top layer to the ground. On the other side, the heat generated by the IC corresponds to the injection of electric current into those nodes that are grouped together in a floorplan block. Figure 2.6 highlights the horizontal section of the silicon wafer used to build a single 2D IC where the floorplan blocks are placed. 3D ICs can be easily modeled stacking and replicating this structure in the vertical direction.

The RC circuit obtained from the structure of the IC is represented by a set of ordinary differential equations corresponding to Equation (2.5) previously introduced in Section 2.5.1. Using the terms defined in the previous Equations (2.3) and (2.4), the conductance g of each resistor and the capacitance c of the thermal

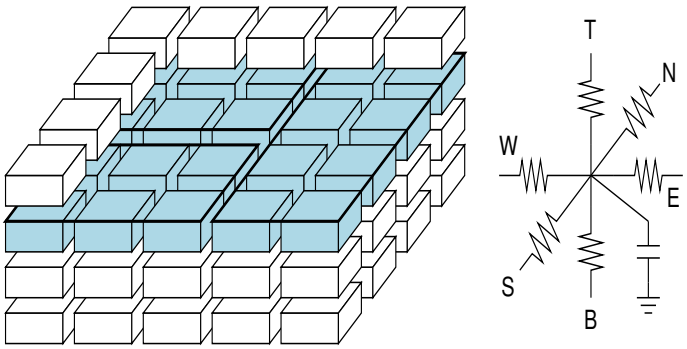


Figure 2.6: Division of the IC into thermal/electrical cells

cell are calculated as follows:

$$g_T = g_B = k\frac{\Delta_x \Delta_y}{\frac{\Delta_z}{2}}, \quad g_N = g_S = k\frac{\Delta_y \Delta_z}{\frac{\Delta_x}{2}},$$

$$g_E = g_W = k\frac{\Delta_x \Delta_z}{\frac{\Delta_y}{2}}, \qquad c = C_v\, V.$$

The system is solved via numerical integration using discrete time h and the backward Euler method. The solution at the $(n+1)^{th}$ time point is written as follows:

$$T(t_{n+1}) = PT(t_n) + QU(t_{n+1}), \tag{2.7}$$

where,

$$P = \left(A + \tfrac{C}{h}\right)^{-1} \cdot \tfrac{C}{h} \quad \text{and} \quad Q = \left(A + \tfrac{C}{h}\right)^{-1}.$$

This thermal model has been implemented and released by the authors of [138] as an open source library called 3D-ICE. Despite the reduced simulation times obtained running the 3D-ICE emulator, the integration of such tool in a thermal-aware floorplanner poses a major drawback as the simulation of large architectures is a highly consuming task in terms of time. Hence, a model that allows faster thermal simulations is needed.

Table 2.4 shows the most relevant thermal properties of the material used in this

Si linear thermal conductivity	295 W/(mK)
Si quadratic thermal conductivity	-0.491 W/(mK^2)
SiO_2 thermal conductivity	1.38 W/(μmK)
Si specific heat	1.628 x 10^6 J/m^3K
SiO_2 specific heat	4.180 x 10^6 J/m^3K

Table 2.4: Thermal properties of materials.

model. Note that these values were set to mimic the experimental conditions of [37], a work by D. Cuesta.

Neural Network Thermal Model

In [139] the authors propose to accelerate the thermal simulation of 2D/3D ICs using a Neural Network. However, this efficient thermal modeling approach has never been used as guiding cost function in a floorplanning tool. Neural Networks are multi-input multi-output operators that can be trained to mimic the behavior of any mathematical function from some test data [108]. Therefore, to compute the temperature profile of a 3D IC as needed by a floorplanner, the Neural Network must be trained to reproduce the outputs of Equation (2.7). Once the training is done, the NN can be used as a thermal simulator instead of 3D-ICE. The main advantage of using the NN-based simulator relies on the high degree of parallelization behind a Neural Network which can benefit from execution in massively parallel architectures such as GPUs.

In the Neural Network Thermal Model (NNTM), each neuron computes the temperature of a cell (or node) in the layers of the IC where the floorplanner places the blocks. Consequently, the number of neurons (outputs) in the Neural Network would equal the number of thermal cells in the active layers of the corresponding IC simulated with 3D-ICE. This set of neurons forms the unique layer of the NN that connects the input of the system directly to its output. To predict the future thermal state $T(t_{n+1})$, i.e. the output, all the neurons receive as input the actual thermal state $T(t_n)$ of the thermal cells in the active layers and their power consumption $U(t_{n+1})$ received from the floorplans. Therefore, the entire network can be described

with the following equation:

$$T(t_{n+1}) = W \cdot \begin{bmatrix} T(t_n) \\ U(t_{n+1}) \end{bmatrix}, \tag{2.8}$$

where the matrix W stores the weights for each neuron (one neuron per row). The values of the weights for any given neuron represent how much the temperature or the power of any surrounding thermal cell contribute as a scalar factor to the variation of its temperature. The matrix-vector multiplication in Equation (2.8) can be implemented in parallel and performs efficiently on GPU. Therefore, the NNTM is suitable for thermal-aware floorplanners as it allows fast thermal simulations of 3D ICs.

The computational and memory complexity of these Neural Networks can be significantly reduced through the introduction of the *proximity*, a parameter that defines a horizontal squared surface around each neuron. Given the diffusive nature of heat flow in an IC, much of the heat flows vertically upwards from the source to the ambient, following the path of least resistance. Hence, there is very little heat flow/interaction between thermal cells that are far apart within the same layer and the connection of individual neurons in the network can be limited to a reduced set of neighbors lying within the area defined by the proximity. Figure 2.7 shows the connections for a single neuron when the proximity parameter is set to one, i.e. only the closest cells contribute to its output.

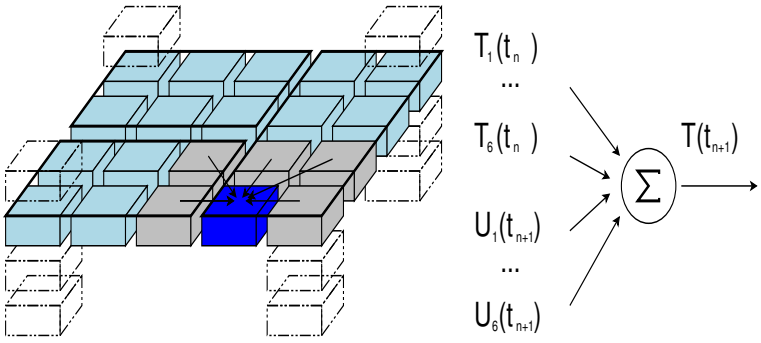


Figure 2.7: Input/output connections of a single neuron in the network

The training phase of the Neural Network essentially consists in finding the

correct set of weights for each neuron. This process must ensure that the Neural Network is capable of predicting temperatures for all possible power inputs and temperature states. As proposed in [139], a training set feeding the 3D-ICE simulator with maps with a random power distribution must be generated. This way, every neuron will be trained to respond correctly for all the possible distributions of power over the surface where blocks are placed. This set up is crucial since the configuration of the floorplan will not be known a priori and the floorplanner must evaluate the steady state thermal profile due to every possible placement of the IC blocks. Figure 2.8 illustrates how a different layout of the floorplan can influence the input fed to a neuron: on the left, the highlighted neuron receives only the power values coming from a single block, with a known power density. On the right, a change in the floorplan will assign the connections of the neuron to two different IC blocks. If the Neural Network is trained with the real power maps matching the configuration on the left and then run with the situation drawn on the right, then the highlighted neuron will give a wrong response because it has not been trained on that case.

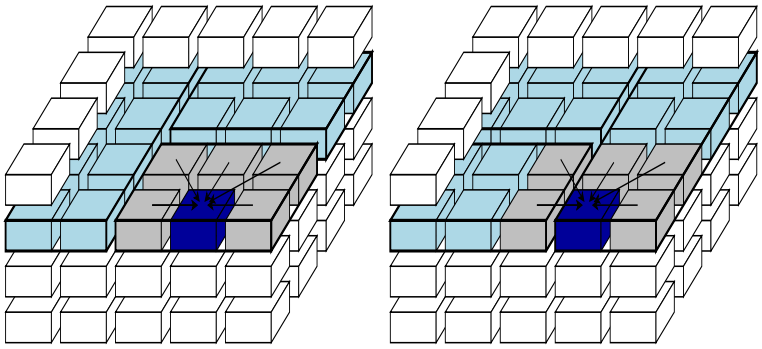


Figure 2.8: The effect of a change in the floorplan on the input of a neuron

The training set will be finally made of the following set of power and thermal maps:

$$\langle \underbrace{T(t_0), U(t_1)}_{Input}, \underbrace{T(t_1)}_{Output} \rangle, \ \langle \underbrace{T(t_1), U(t_2)}_{Input}, \underbrace{T(t_2)}_{Output} \rangle, \ \ldots$$

where the values $U_{i,j}(t_n)$ will be random values bounded by the power consumption of the IC block with the highest power density. This will make the resulting Neural Network independent from the placement of the blocks. During each training iteration, the inputs are given to the neurons and the outputs from the Neural Network

and the target output from 3D-ICE are compared. The weights of the corresponding neurons are then updated according to the error incurred. These iterations are repeated until convergence is reached based on some predefined error tolerance. Once the training is finished, these weights can be stored and reused in future simulations.

This thermal model had never been included in a thermal optimization process. We will demonstrate its suitability for thermal-aware floorplanning in Section 5.4.

Approximated Thermal Model

Most of the existing proposals consider approximated models to guide the thermal optimization of 2D and 3D architectures. These simplified models generally present a low computational cost translated into a fast runtime, necessary for iterative optimization approaches in which a large number of thermal simulations are carried out. For example, in [38] it is assumed that in optimal solutions the hottest blocks are placed as far as possible in the 3D stack. Under such assumption, the temperature of the hottest blocks will be proportional to their own power density, mainly because adjacent blocks will act as heat sinks. Thus, given a 3D stack discretized in N blocks and given a block i, its temperature is modeled as follows:

$$T_i = \sum_{j=1}^{N} \mathbf{D}_{i,j} P_j \approx \mathbf{D}_{i,i} P_i \propto p_i \tag{2.9}$$

where the matrix D stores the euclidean distances between the center of the blocks in the floorplans and p_i is the normalized power density of the referred block i. Figure 2.9 shows the discretization of the volume of a single die of the 3D stack into parallelepipeds according to the dimensions of the blocks in the floorplan.

Equation (2.9) states that the power consumed within a block will be proportional to its final thermal state, without considering the volume and the mass of the material and the distances between blocks. However, this approximation is only valid if the hottest blocks are placed far from each other in the 3D stack. Thus, the minimization of the metric

$$\text{Min. } \hat{T} = \sum_{j<i \in 1 \ldots N} \frac{p_i \cdot p_j}{\sqrt{dx_{ij}^2 + dy_{ij}^2 + dz_{ij}^2}} \tag{2.10}$$

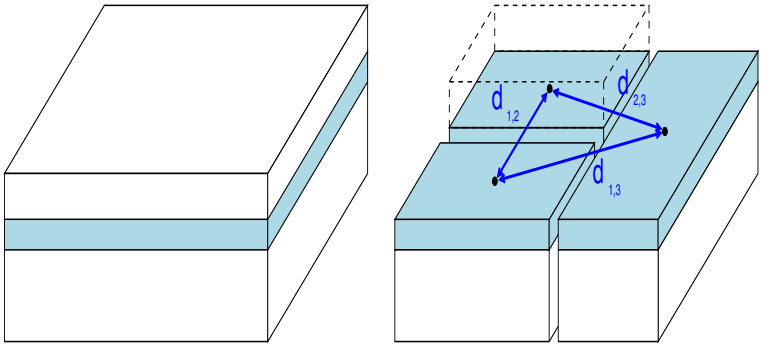


Figure 2.9: Approx thermal model with the 3D stack discretized in blocks

will drive the optimization process towards a minimum thermal state. Barely speaking, this metric provides an insight on how close the hottest blocks are. In the referred work [38], it is demonstrated that targeting the minimization of $\hat{T}$ leads to a maximum temperature reduction. Note that $\hat{T}$ in Equation (2.10) is not a temperature but an approximated metric: to obtain the distribution of temperatures corresponding to the placement of the blocks and validate the final solutions, a thermal simulator must be used.

2.6 Conclusions

In this work, we target the design of 3D MPSoCs following the Entire Cores and Memories design approach. As explained in Chapter 1, MPSoCs allow to keep the performance enhancement trend going. 3D designs have the potential to reduce the overall wire length of such architectures obtaining an increased performance. Also, as showed in this chapter, these systems are the only ones that would fully benefit from a 3D Integration technology available in the short term and from the almost full reuse of existing 2D designs and tools. In fact, the components forming these systems (mainly cores and memories), can be treated as IP blocks and systems can be assembled by choosing a mix of blocks to stack together.

Some of the critical design challenges imposed by 3D Integration can be faced with existing tools. In particular, thermal-aware floorplanning and management and a tool infrastructure to support them, are considered essential for designing

high-power systems such as the targeted 3D MPSoCs [52]. In fact, power density increases linearly with the number of layers leading to significant thermal issues. 3D organizations provide additional degrees of freedom that can be exploited by processor architects, circuit designers, and layout engineers. Thermal-aware design is a significant challenge but, as we will show in this work, it is by no means, an insurmountable obstacle for the adoption of 3D Technology [102].

Chapter 3

Floorplanning Representations

Floorplanning is a crucial step in Very Large Scale Integration (VLSI) physical design. Traditionally, its main objective has been to minimize the total area required to place all of the functional blocks on a chip. Floorplanning techniques are based on different representations that determine the cardinality of the solution space. This means that the chosen representation and algorithm will have a great impact on the obtained solutions. In fact,

- not all the possible solutions to a given problem are reachable with all the representations
- different representations might lead to different solutions

In addition, the analysis of the computational complexity of the problem depends on the employed representation. For instance, in [17] Berntsson *et al.* propose the use of a slicing structure representation, in which case, the floorplanning problem corresponds to a generalization of the quadratic assignment problem, considered an NP-hard problem [58].

Several Mixed Integer Linear Programming (MILP) approaches with relaxed constraints have been proposed to attack the floorplanning problem [51], [95], and [37]. However, the goal is not necessarily finding the global optimum of the problem, but rather to obtain a good enough solution in still a short time, specially in a multi-objective optimization context. Moreover, when dealing with thermal constraints,

the (linear) thermal model must be added to the topological relations and the resultant algorithm becomes highly time consuming when the problem size increases.

Initial heuristic approaches to this problem can be divided into two general categories: constructive and iterative. The first approach starts from a seed module, and adds modules to the floorplan until all modules have been placed. The latter starts from one or more initial floorplans that are iteratively improved, generally with Simulated Annealing (SA) or Evolutionary Algorithms (EAs). In this chapter, we focus on the iterative class as it is found in most of the state-of-the-art proposals. In fact, early works tackled floorplan design with SA [155], [84], and GAs [27], and [137]. These proposals inspired a number of works in the 2000s targeting the floorplanning problem at a logic circuit level. In fact the Microelectronics Center of North Carolina (MCNC) and Gigascale Systems Research Center (GSRC) benchmarks have been widely used to compare different works such as [146], [97], [1], [141], [17], [31], [135], [142], [69], [143], [64], [22], [100], [23], [95], and [118]. In a similar way, the IBM-PLACE benchmark [13] which is composed of standard cell circuits is used in recent works such as [30], or [29].

These validated proposals have been reported successful for the floorplanning of 2D and 3D architectures targeting area and/or wire length. However, the targeted 3D MPSoCs present different restrictions such as a fixed-outline constraint or severe thermal hotspots typically not taken into account in the referred works. Recent works approach the thermal management of 3D MPSoCs assuming a regular structure of the components [160] and [63]. However, rearranged floorplans might mitigate the effect of severe hotspots present in strictly regular configurations [91]. It is therefore necessary to adapt and study the suitability of the existing proposals for the 3D thermal-aware floorplanning of the targeted platforms.

The most relevant representations are explained in the first section of this chapter, namely Combined Bucket and 2D Array, Double-Tree and Sequence, Sequence Pair, and Generalized Polish Expression. In the second section a comparison of these state-of-the art floorplanners performing for the optimization of 3D MPSoCs is proposed.

3.1 Common Representations

3.1.1 Combined Bucket and 2D Array

Combined Bucket and 2D Array (CBA) representation emerges from the aspiration of providing a solution that avoids the unnecessary overhead of "true 3D" solutions on one side, as the number of layers is fixed in real world applications, and the possibility of introducing effective z-axis perturbations that would not be limited by the 2D-array representation on the other. The representation was first proposed by Cong *et al.* in [31], and its main idea consists in:

- proposing a bucket structure to encode z-axis neighboring
- using Transitive Closure Graphs (TCG) [98] to encode a 2D floorplan on each layer

This proposal is compatible with other 2D representations, yet TCG presents several advantages, which will be explained in detail in the following text.

The TCG representation depicts the geometric relations between different modules using two graphs, a horizontal transitive closure graph C_h, and a vertical transitive closure graph C_v, defined in the following manner. Two non-overlapped modules are horizontally related if one is on the left side of the other, and their projections on y-axis overlap. In the same way, two modules are vertically related if one is positioned below the other, and their x-axis projections overlap. It is important to notice that two modules can be in either horizontal or vertical relation, as in the opposite case they would overlap. A diagonal relation between modules is possible as well, and it is defined for two non-overlapping modules, where one is on the left side of the other, and their projections on both x- and y-axis do not overlap. In order to simplify the codification, diagonal relations are mapped to vertical or horizontal relations. This is depicted in Figure 3.1; the weight of each node corresponds whether to its horizontal length, or to its height in the vertical graph. Given a TCG, the corresponding placement of each module is obtained by performing the longest-path algorithm [134] which, in the case of direct acyclic graphs, can be solved in linear time using dynamic programming. In order to facilitate this calculation, two nodes are added to both the horizontal and vertical graphs: the source node n_s with weight

0, connected to each node with fan-in equal to 0, and the sink node n_t also with weight 0 and connected to each node with fan-out equal to 0. Thus, the coordinate (x_i, y_i) of the upper right corner of a module n_i is given with $(L_h(n_i), L_v(n_i))$, where $L_h(n_i)$ and $L_v(n_i)$ are the longest paths between the node and the corresponding source node. In the same way, the area of the placement is determined by the product $L_h(n_t)L_v(n_t)$, where n_t is the sink node. The important properties of the TCG are [98]:

1. C_h and C_v are acyclic.

2. Each pair of nodes must be connected by exactly one edge either in C_h or in C_v.

3. The transitive closure of $C_h(C_v)$ is $C_h(C_v)$ itself.

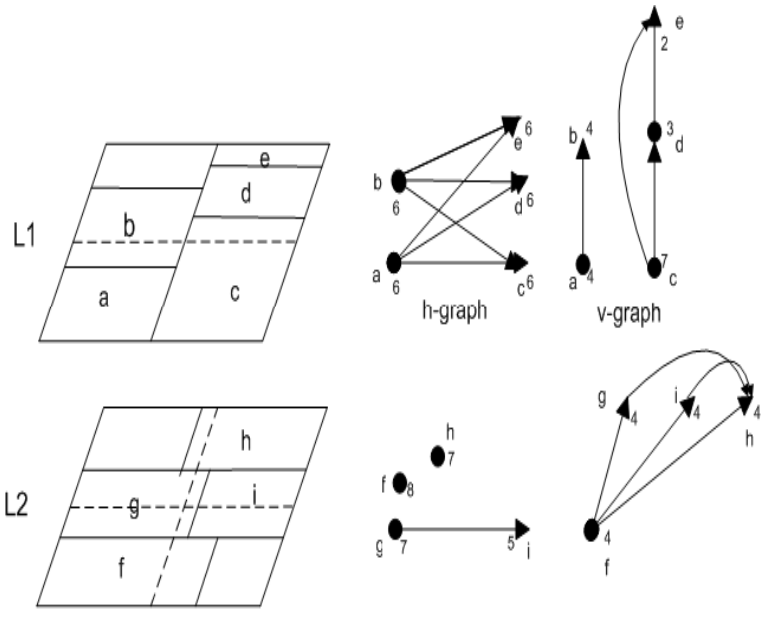


Figure 3.1: An example of two-layer floorplan, its corresponding TCG for each layer.

This representation offers a number of advantages, namely it does not need to construct additional constraint graphs for the cost evaluation during packing, which implies faster running time. Furthermore, it supports an incremental update during operations, and keeps the information of boundary modules as well as the shape and the relative position of modules. Moreover, the geometric relation between modules

is transparent not only to the TCG representation, but also to its operations, providing a faster convergence to a desired solution and a placement with position constraints.

Regarding the bucket structure, the size of the bucket ($n \times m$) has to be defined first. For each bucket i, the indexes of the modules that intersect with the bucket are stored in $IB(i)$, no matter on which layer the modules are located. At the same time, each module j stores the indexes of the buckets that overlap with it ($IBT(j)$). Table 3.1 shows the 2×2 bucket structure imposed to two layers, where the indexes of the blocks are listed in the buckets with which they overlap. This facilitates the definition of the efficient z-axis perturbations given in the following.

b,g,h	c,d,e,i,h,g
a,b,f,g	c,f,g,i

Table 3.1: The corresponding 2×2 bucket list.

Before explaining the perturbations, we should define two terms. The first term is reduction edge. An edge (n_i, n_j) is said to be a reduction edge if it does not exist another path from n_i to n_j, but the edge (n_i, n_j) itself. In the opposite case, the edge is a closure edge. The second term is the z-axis neighbor, based on the bucket structure. In order to find the z-axis neighbor of a given module j, $zn(j)$, we first define a set of modules that are close to module j. This is the set of all the modules that share at least one bucket with j, formally written in the following way:

$$B(j) = \bigcup_{i \in IBT(j)} IB(i) \tag{3.1}$$

$zn(j)$ is defined as the block k in $B(j)$ with the minimum neighboring cost

$$zc(k, j) = \alpha |A_k - A_j| + \beta |x_k - x_j| + |y_k - y_j|), \tag{3.2}$$

where A_k and A_j are the corresponding levels, and α, β are coefficients lower than 1. If $zc(j, k)$ is small, then blocks j and k probably have similar area and are close to each other, or even overlap along z-axis. The notion of z-axis neighbor will be used

in local interlayer perturbations, as it will cause small changes to the floorplan and the cost function.

Seven operations are used for solution perturbation. A more detailed explanation can be found in [98] and [31].

1. *Rotation*: rotates the block; in essence, it consists of exchanging the weights of a node n_i in horizontal C_h and vertical C_v graph.

2. *Swap*: swaps two modules in one layer; it is performed by exchanging two nodes in both C_h and C_v .

3. *Reverse*: reverses the direction of a reduction edge, which further corresponds to changing the geometric relationship of two modules. If the edge was a closure one, the result of reversing would be a formed cycle, which means that the graph would not be acyclic, and thus it would not be a TCG.

4. *Move*: moves a reduction edge in a TCG to the other, which corresponds to switching the geometric relation of the two modules between a horizontal geometric relation and a vertical one, or vice versa. The edge has to be a reduction edge for the same reason as before.

5. *Interlayer Swap*: swaps two modules at different layers and between their corresponding graphs.

6. $Z-$*neighbor Swap*: swaps a module and its z-axis neighbor and between their corresponding graphs.

7. $Z-$*neighbor Move*: moves a module to the layer of its z-neighbor, at either its top or right side and updates the corresponding graphs.

3.1.2 Double-Tree and Sequence

Several authors represent floorplan configurations using ordered trees (O-trees) to solve the placement of components in 2D [141] and in 3D chips [17]. O-trees build a non slicing representation claimed to cover all optimal floorplans and, at the same time, explore a smaller search space in relation to other non slicing representations.

It is important to note that in this case, optimality of the solutions refers only to the area of the represented configurations.

We consider the floorplan representation inspired in O-trees described in [57], called Double-Tree and Sequence (DTS). Typically, SA algorithms are applied to optimize 3D platforms with this representation.

Three elements are required to describe a DTS 3D configuration:

1. *x-tree*, a tree describing the relation between components along the x-axis
2. *y-tree*, an analogous tree describing the y-axis
3. *z-order*, a sequence that describes the order along the vertical z-axis

As stated in [57],

- *x-tree* is a rooted multiway tree, where each node corresponds to a rectangular box, except that the root node of the tree represents a pseudorectangular box whose x-size is zero, but the y- and z-sizes are infinite. The *x-tree* imposes the following constraint: the x coordinate of the left plane of the box corresponding to a given node must be the same as that of the right plane of the box corresponding to its parent node

In a similar way,

- *y-tree* is a rooted multiway tree, where each node corresponds to a rectangular box, except that the root node of the tree represents a pseudorectangular box whose y-size is zero, but the x- and z-sizes are infinite. The *y-tree* imposes the following constraint: the y coordinate of the front plane of the box corresponding to a given node must be the same as that of the rear plane of the box corresponding to its parent node

Figure 3.2 illustrates this definition. In this example, the floorplan consists of two layers: layer 0, with components a, b and c; and layer 1, with components d and e. From the x-axis point of view, the root node of *x-tree*, represented as a black-colored node, is the first element in the axis and its x coordinate is set to 0. Given that the

root has zero-size, its children will have their x coordinate set to 0 as well, because the coordinate of an element is obtained by adding the coordinate of the parent to the size of the parent. As seen in the example, a, c and e, are located at 0. Hence, they are children of the root. Note that the left-to-right order does not matter here, just the parent-children relationship. Now, from the x-axis perspective, b is after a and d is after e. Then, b is the child of a and d is the child of e in the tree. The same applies to the *y-tree* and the y-axis coordinates. Notice that c is the child of a because, according to the *x-tree*, the x coordinate of c is the same than in a, while b has a higher x coordinate.

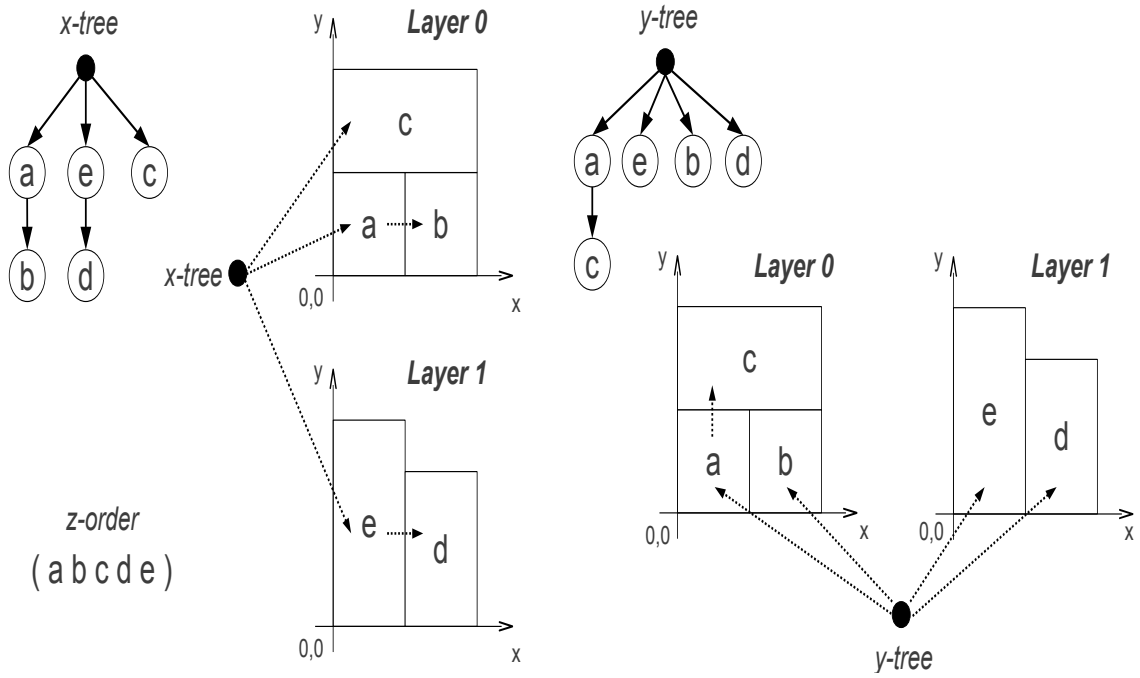


Figure 3.2: *x-tree*, *y-tree* and *z-order* for a floorplan configuration of five components (a to e) in two layers.

The *z-order* sequence is taken into account to establish the z-axis relationship. If two nodes with the same z coordinate overlap, the latter in the *z-order* sequence is moved to the next layer (notice that all components must have a z coordinate lower or equal than the number of layers).

In order to decode a DTS representation, the trees are traversed to obtain coor-

dinates for the x- and y-axis. The *z-order* sequence is then processed from left to right to determine the vertical relationship.

In the example shown in Figure 3.2, the x-y planes formed by a, b and c do not intersect between them. They all have z coordinate equal to 0. However, the plane formed by d intersects with b and c. Then, d takes the following z coordinate, which is 1. The plane given by e is compared to the nodes at the current coordinate. Then, as e does not intersect with d, e also takes z coordinate 1.

In [57], two operations are implemented to perturb a DTS individual:

1. Moving a node in the *x-tree* or in the *y-tree*;
2. Swapping two elements in the *x-tree*, *y-tree* or *z-order*.

Each time a perturbation is needed, a node is moved either in the *x-tree* or in the *y-tree*, and a swapping is performed in one of the three structures.

3.1.3 Sequence Pair

Sequence Pair (SP) is another popular codification employed in VLSI layout design. It was first introduced in [117], where a SA algorithm aims to minimize the wire length of the connections between the different components. Later, Okada *et al.* employed this representation for thermal aware floorplanning in [120]. A Genetic Algorithm (GA) using this codification has also been engineered [49]. This last proposal allows the rotation of the components and was tested on the MCNC benchmark for minimum packing area only. Finally, a solution space reduction was presented in [158].

SP codification requires every component to be uniquely associated to an identifier i. For the sake of simplicity,

- Let i be a positive integer
- and let $\mathcal{I}$ be the set of all the components that are to be placed in a layout

The SP representation of a floorplan for such a layout consists of two sequences, Γ^- and Γ^+, in which all the $i \in \mathcal{I}$ appear only once. In other words, both Γ^- and Γ^+ are permutations of all the elements in $\mathcal{I}$.

The extension of SP for 3D stacked chips is straightforward if every component must be completely embedded in only one layer. Hence, the 3D SP codification is a list of pairs (Γ_j^-, Γ_j^+), where j refers to the layer. Thus, from now on, we no longer refer to the layout but rather to the layers of the 3D stacked chip.

The SP decodification is the process of retrieving the coordinates (x_i, y_i, z_i) of the left-bottom-back corner of every component $i \in \mathcal{I}$, with respect to the left-bottom-back corner of the layer, given Γ^- and Γ^+. This process consists of two steps: the construction of the so-called *constraint directed acyclic graphs* G_H and G_V and their interpretation in order to obtain the coordinates of each component.

Both G_H and G_V are defined over the set of vertices $\mathcal{I}$ and their edges are obtained with the following procedure (see Figure 3.3):

1. Write Γ^+ in the first row of a table. Then, in the next row, write Γ^-. Finally construct $\mid \Gamma^- \mid$ empty rows of decreasing size and write Γ^- in the diagonal cells.

2. Pick the first element in Γ^- and locate it in Γ^+. Let us refer to the element in Γ^- as *anchor*. Then, form the set containing all the elements in Γ^+ to its right.

3. Pick the first element of the set and locate it in Γ^-. If this position is to the right of the anchor then assign an H to the cell $(anchor, element)$ in the half-table.

4. Repeat the last step with every element in the block.

5. Repeat the whole procedure with every element in Γ^-

6. Two vertices i, j are connected in G_H if the cell (i,j) contains an H while G_V has an edge connecting i and j if the cell is empty.

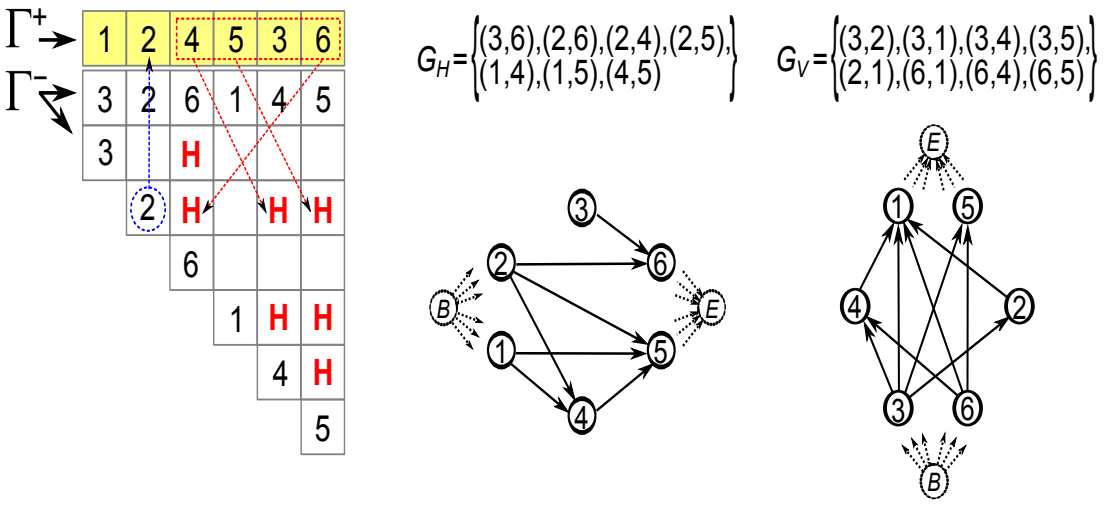


Figure 3.3: Obtention of G_H and G_V from the sets $(\Gamma^+, \Gamma^-) = (124536, 326145)$. The edges of G_H correspond to the cells marked with H while the edges of G_V correspond to the cells in blank. The edges connecting *Begin*(B) and *End*(E) with the rest of vertices are not written for clearness.

In the example shown in Figure 3.3, the SP is given by $(\Gamma^+, \Gamma^-) = (124536, 326145)$. The first anchor is 3. Its corresponding set in Γ^+ is {6}, which is to the right of the anchor. Hence (3,6) is an edge of G_H. The second anchor is 2. Its corresponding block in Γ^+ is {4,5,3,6}, but only 4, 5 and 6 are to its right. Hence the edges (2,4), (2,5), and (2,6) are added to G_H, while the cell (2,1) remains empty. This case is highlighted in the figure. The procedure continues with the remaining elements of Γ^-. The construction of G_V is then straightforward as the edges correspond to the blank cells of the table. Finally *Begin* and *End* vertices are added both in G_H and G_V connecting with the rest of vertices in the graphs.

Regarding the interpretation of the graphs, a vertex connecting p and q in G_H forces to place component p left to q. Similarly, if the edge (p, q) is in G_V, then component p is must be placed below q. In addition the vertex weight is the component's width or length for G_H or G_V respectively. Thus, the weight of the longest path within G_H from the vertex B to vertex p is the x coordinate of component p. Likewise, the y coordinate of component p is obtained with G_V.

According to the features of SP codification, the following operators have been used:

1. Swap adjacent elements inside a given Γ^+ or Γ^-.
2. Swap components in the same layer.
3. Swap two entire layers.
4. Move one component *upwards* or *downwards*.

Table 3.2 illustrates these operators in a synthetic example of a six layers stacked chip, as if all the operators were applied at the same time on different layers.

Layer j	Original Floorplan $\{\Gamma_j^+, \Gamma_j^-\}$		New Floorplan $\{\Gamma_j^+, \Gamma_j^-\}$
1	$\{(1,2,3),(2,1,3)\}$		$\{(2,1,3),(2,1,3)\}$
2	$\{(4,5,6),(5,6,4)\}$		$\{(5,4,6),(4,6,5)\}$
3	$\{(7,8,9),(9,8,7)\}$	$\Longrightarrow$	$\{(10,11,12),(10,11,12)\}$
4	$\{(10,11,12),(10,11,12)\}$		$\{(7,8,9),(9,8,7)\}$
5	$\{(13,14,15),(15,13,14)\}$		$\{(14,15),(15,14)\}$
6	$\{(16,17,18),(16,18,17)\}$		$\{(13,16,17,18),(16,18,13,17)\}$

Table 3.2: Example of SA operators in SP. (1) swapping 1 and 2 in Γ_1^+, (2) swapping 4 and 5 in layer 2, (3) swapping layers 3 and 4, and (4) moving 13 downwards.

3.1.4 Generalized Polish Expression in 3D (GPE)

This 3D representation was first proposed in [97]. We follow the example presented in the referred paper to briefly describe the proposal. Figure 3.4 shows the binary tree for the Polish expression "3 1 6 8 Z H Z 2 7 Z V 5 4 H V". In this case, a floorplan is encoded in normalized Polish expressions using the symbols H, V, and Z to represent horizontal, vertical, and lateral cuts respectively. The integers $1 \ldots n$ identify the blocks that must be placed (block ID).

It is necessary to break down the 3D slicing structure to evaluate the floorplan. A splicer algorithm (as referred in [17]) takes as inputs a 3D floorplan and the maximum number of layers, and returns a slicing structure for each layer. The trees

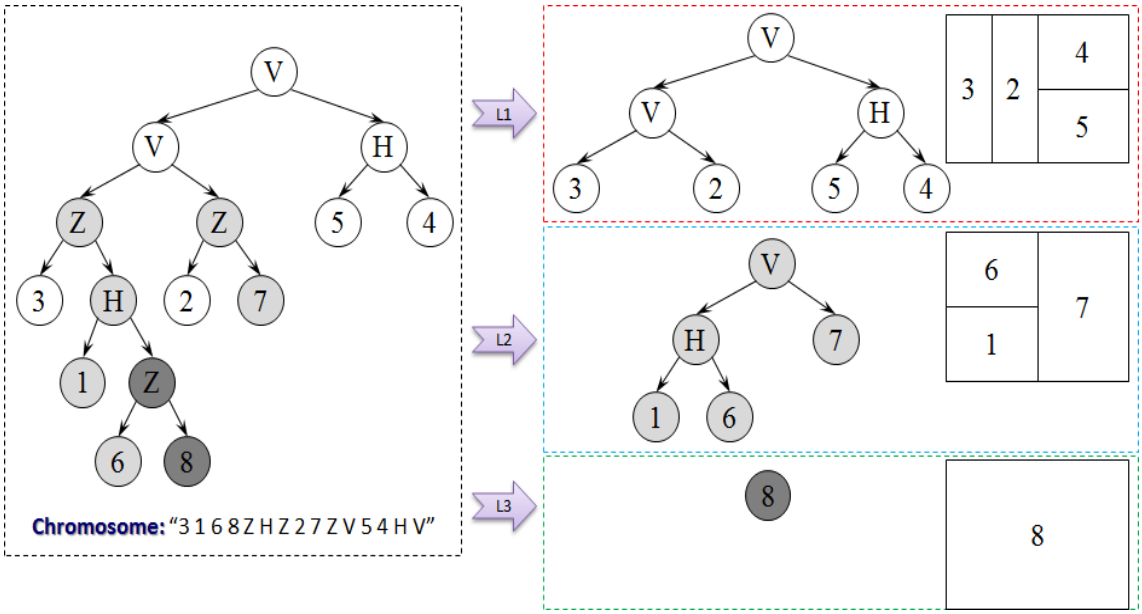


Figure 3.4: The 3D floorplan tree for "3 1 6 8 Z H Z 2 7 Z V 5 4 H V", its three 2D layers, and the three slicing floorplan layers (right).

on the right side of Figure 3.4 show the Polish expression divided into the three layers. The splicer algorithm examines the binary tree for the 3D slicing floorplan, and removes the Z nodes, leaving a 2D slicing tree with only V and H nodes. All the Z nodes in the current layer are replaced with their left child while their right child is inserted into the next layer. Since the maximum number of layers is fixed, the function wraps around and starts reinserting nodes in the first layer when it reaches the top layer. More details about the splicer algorithm can be found in [17]. Figure 3.4 also shows the corresponding slicing floorplans, after applying the splicer algorithm.

The genome consists of an array of records, each of which corresponds to a block in the problem, composed of three fields: (1) a block ID, which identifies the block from the dataset, (2) a chain type ID that can be set to one of the following three operator chains (1=VHVHV..., 2=HVHVHV..., and 3=ZHZV...), and (3) a chain field length, which defines the maximum length of the associated operator chain.

There are three mutation operators that modify the genome:

1. Swap the positions of two block IDs.

2. Rotate a randomly chosen block 90 degrees.

3. Change the chain type (for more details see [17]).

The presented representations have been reported successful in the context of the optimization of 2D and 3D logic circuits. However, the critical thermal restrictions imposed by large 3D MPSoCs make necessary the validation of these techniques in the new scenario. In fact, managing thermal constraints is considered one of the key challenges for 3D integration. The general idea is that a strategically rearranged floorplan might mitigate the effect of severe hot spots present in strictly regular configurations [91]. It is therefore necessary to adapt and study the suitability of the existing proposals for the thermal-aware floorplanning of 3D MPSoCs.

3.2 Comparative Study

In this section we report the comparative study of the state-of-the-art floorplanners presented in [40]. The four representations explained in Section 3.1 have been used to perform the thermal-aware floorplanning of large 3D MPSoCs in which **both wire length and temperature are targeted**. Such contribution is a valuable asset as, to the best of our knowledge, the suitability of the compared techniques for targeting state-of-the-art 3D MPSoCs had never been studied before. The reader is referred to Chapter 4 for an introduction to multi-objective optimization. The details of the algorithms and adaptation of these techniques are given below.

3.2.1 Adaptation of the State-of-the-art Floorplanners

Combined Bucket and 2D Array

A **Simulated Annealing** algorithm is used in this case. As the SA engine iterates, a new CBA representation is obtained through perturbations. Each time a floorplan is generated, the corresponding cost function is calculated, i.e. a **weighted sum of temperature and wire length**. The SA engine generates more global

operations in the beginning, i.e. rotation, swap and interlayer swap, while the temperature of the annealing is high. As it decreases, more local operations are generated, i.e. move, reverse, z-neighbor swap and z-neighbor move, this way the perturbations are more likely to be accepted. The perturbed layers and modules are picked randomly. In the case of starting with an unfeasible solution, critical layers, i.e. those violating the higher number of boundary constraints, are more likely to be elected. Furthermore, when the managed candidate solution is unfeasible, the cost function is penalized according to the exceeded area per layer. The penalization function is necessary as it provides fast convergence to a feasible solution.

Double-Tree and Sequence

A **Simulated Annealing** algorithm is also used in this case. The cost function is a **weighted sum of temperature and wire length** as well. As for the previous proposal, a penalization function is defined for this codification in order to favor the feasibility of the solutions. However, in this case, the penalization is global to the floorplan, because the per-layer calculation requires a higher computational effort when using the DTS representation.

Sequence Pair

Alike CBA and DTS, a **standard SA algorithm** has been employed guided by a **weighted sum of temperature and wire length**. In each iteration of the SA, only one operator is randomly selected and applied. Feasibility is enhanced through the same penalty function defined for the DTS codification. Finally, orientation of blocks is fixed so they can not be rotated in order to fit better.

Generalized Polish Expression

The classic GA engine used in [17] has been replaced with **NSGA-II** [46], a well known multi-objective evolutionary algorithm. Comparatively, NSGA-II obtains better results than the classic approach. The phenotype of the candidate solutions (a normalized Polish expression) is constructed as proposed in [17]. The genome is order based, thus a permutation-based cycle crossover (CX) [109] is employed.

Regarding GPE parameters, the population size is fixed to 100, and probabilities of mutation and crossover to the inverse of total number of blocks and 0.9, respectively (as recommended in [46]). In this case, we approach the optimization in **two different ways** targeting:

1. **A weighted sum of temperature and wire length** (marked as GPE_f in the following)

2. **Temperature and wire length as two separate objectives** (GPE in the following)

3.2.2 Experimental Setup

Benchmarks

The studied benchmarks are inspired on the Niagara2 [110] and Niagara3 [33] architectures. These architectures include SPARC cores, fabricated in a Taiwan Semiconductor Manufacturing Company (TSMC) 90nm technology [28] and exhibiting high power density. Power6 cores are also included with a ratio of 3 SPARC / 1 Power6. These heterogeneous architectures are representative of the future trend in 3D MPSoC design. In this case, the heterogeneous computing power of the integrated processors allows for a more efficient workload distribution. Floorplanners must manage the different power consumption values and areas of the different elements of the architectures to produce thermally optimal solutions.

These power consumption values and areas are given as inputs to the thermal-aware floorplanners. In [121] we find that the power consumption of the SPARC is 4W at 1.4GHz. In the case of the POWER6, 2.6W is the estimated power dissipation [73]. The following areas are considered: $3.24mm^2$ and $1.5mm^2$ for the SPARC and POWER6 respectively (see [121] and [73]). A cell size of $300\mu m$ is fixed. Thus, the length and width of the different components are enlarged to the next multiple of the considered cell size. This way, there is a free perimeter around each component used for wire routing. this area overhead is not necessarily relevant as the area of the targeted architectures is fixed beforehand. The power consumption values and areas of the memories are found with the CACTI software [67].

Figure 3.5 depicts the original two layers, based on Niagara2 and Niagara3. Top and bottom blocks are SPARC (bigger) and Power6 (smaller) cores in the ratio of 3/1. The block at the center is a crossbar while the rest of the blocks are L2 memories and shared memories. These two layers are alternatively repeated in order to build three architectures:

1. *NH48*: A 48 cores architecture composed of 4 layers
2. *NH64*: A 64 cores architecture composed of 5 layers
3. *NH128*: A 128 cores architecture composed of 9 layers

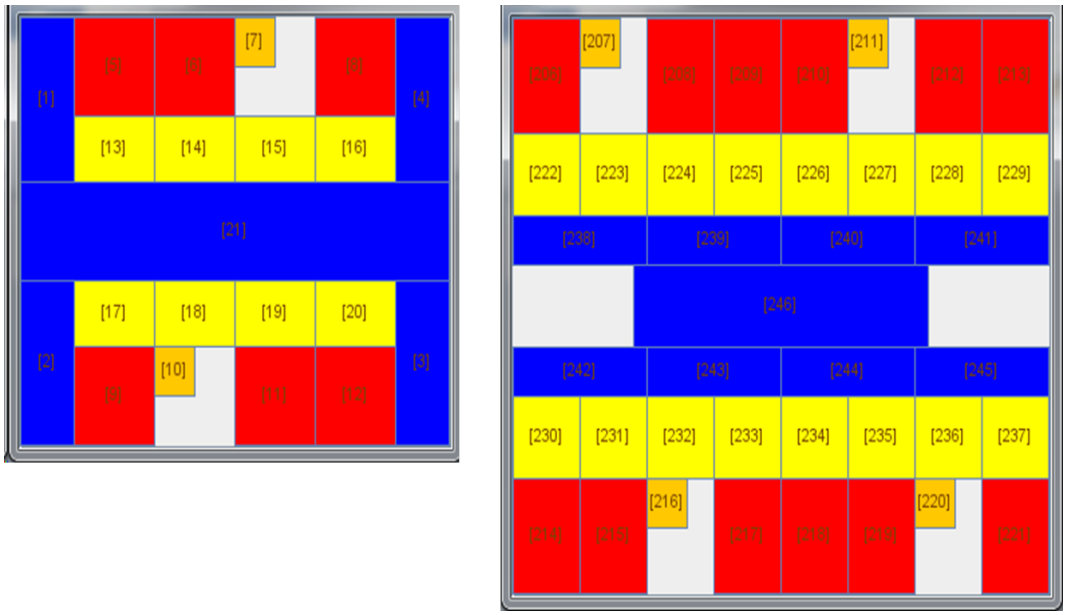


Figure 3.5: Original floorplan layers of the Niagara2 (left) and Niagara3 (right)

The area per layer of the 3D stack is set to the original Niagara 3 area. Note that the area of the Niagara2 layer (left side of Figure 3.5) is therefore increased. Since all the floorplanners being analyzed can place a variable number of cores in each layer, the area and power consumption of the crossbar is scaled according to the maximum number of cores per layer and their required bandwidth. Table 3.3 summarizes the components of these architectures.

Platform	Sparc	Power6	Memories	Crossbars	Total	num. Layers
NH48	36	12	72	4	124	4
NH64	48	16	96	5	165	5
NH128	96	32	191	9	328	9

Table 3.3: Description of the Niagara-based architectures

Fitness Function

The floorplanners will place the functional units that compose the 3D multiprocessor architecture according to the following strategies:

1. Minimizing a weighted sum of wire length and temperature (CBA, DTS, GPE, MFA and SP)

2. Targeting wire length and temperature as two separate objectives (only GPE)

In addition, for each floorplanner and optimization strategy two different experiments are carried out and compared:

1. Starting the optimization from the original configuration of the blocks

2. Starting the optimization from a random configuration of the blocks

Wire Length The wire length is approximated as the Manhattan distance between interconnected blocks and is computed as follows,

$$W = |x_i - x_j| + |y_i - y_j| + |z_i - z_j| \tag{3.3}$$

where x_i, y_i, and z_i are the coordinates of block i. We consider that wires can only be routed parallel to the X or Y axis as generally done in integrated circuits. Therefore, this approximation is exact if any two blocks can be connected by a shortest path. This means that vertical connections between layers are assumed to be optimally placed.

Temperature Temperature is measured through the power consumed by the unitary cells of the chip. The approximated model explained in Section 2.5.2 is employed because of its low computational cost as compared to exact models. For the sake of clarity, we rewrite the employed approximated thermal model:

$$\hat{T} = \sum_{i<j \in 1..n} (dp_i * dp_j)/(d_{ij}) \tag{3.4}$$

where dp is the density power of block i and d_{ij} is the euclidean distance between blocks i and j.

Table 3.4 shows the following characteristics of the three original benchmarks: number of cores, total number of blocks, length (number of cells), width (number of cells), number of layers, approximated wire length in cells according to Eq. (3.3) and approximated thermal metric according to Eq. (3.4). These benchmarks can be downloaded from [126].

Benchmark	N cores	# Blocks	Length	Width	Height	W_r	$\hat{T}_r$
NH48	48	124	40	35	4	1320	146.66
NH64	64	165	40	35	5	1636	250.55
NH128	128	328	40	35	9	2889	879.09

Table 3.4: Reference values for the three benchmarks.

It is worth highlighting that all the three Niagara-like 3D floorplans present short wire length but prohibitive maximum temperatures (389.89 K, 400.34 K and 432.94 K for 48, 64 and 128 cores, respectively [38]). These temperatures are physically unfeasible.

Fixed Time Optimization

All the optimizations (floorplanner + fixed or random seed) were run 10 times. Given that the analyzed codifications and algorithms are different, all experiments were run in the same computer until a wall-clock deadline was reached. This limit was set to:

- 12 hours for the 48 cores architecture
- 48 hours for the 64 cores architecture
- 96 hours for the 128 cores architecture

3.2.3 Results

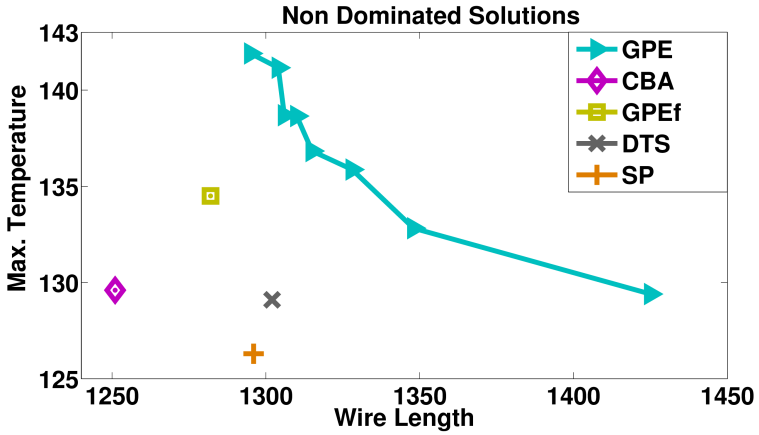
Results are shown in Figure 3.6. The symbols ▷ show the front of nondominated solutions found with the GPE representation. For further details on the concept of domination, the reader is referred to Section 4.2.1. In the case of CBA, GPE_f, DTS, and SP the values showed in the figure correspond to the average W and T obtained over the 10 simulations.

All the approaches improve the original Niagara configurations, which are highly optimized for wire length, but not for temperature. The multi-objective optimization performed with GPE does not reach the thermal optimization achieved with the other proposals. In the 48 cores scenario, CBA and SP obtain the best configurations. DTS and SP are clearly outperformed in the 64 cores scenario while they present the lowest metrics in the case of 128 cores. It is therefore not clear which proposal, if any, is most suitable for the 3D thermal-aware floorplanning problem.

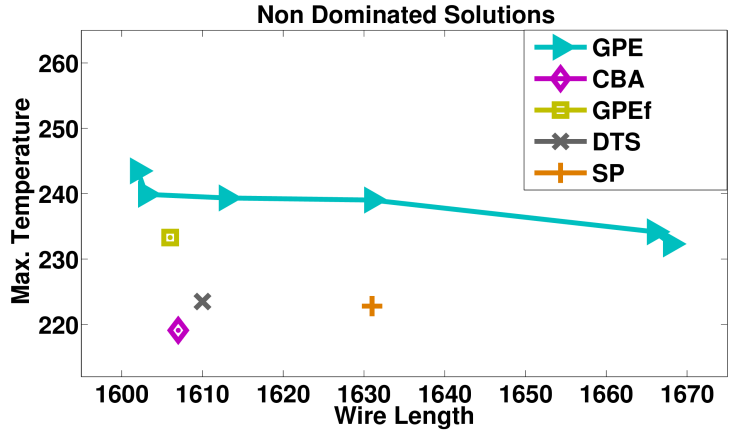
It is worth noting that the results presented here correspond to the execution of CBA, DTS, GPE and SP with a fixed seed, namely the original configurations explained in Section 3.2.2. In fact, these approaches do not reach feasible solutions when initialized with a random seed.

3.3 Conclusions

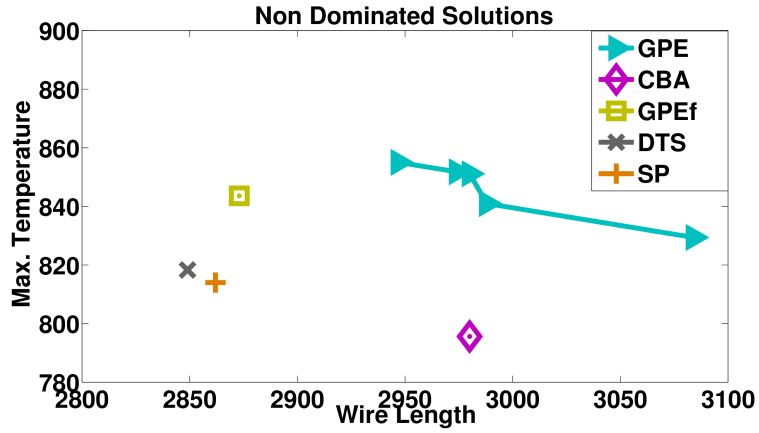
In this chapter, several well-known floorplanning techniques typically employed for the optimization of 2D logic circuits have been adapted to attack the thermal-aware floorplanning of large 3D MPSoCs. Three large 3D architectures based on the Niagara 2 and Niagara 3 have been used as benchmarks to compare the suitability of the different techniques.



(a) 48 cores scenario



(b) 64 cores scenario



(c) 128 cores scenario

Figure 3.6: Optimized solutions found with GPE, CBA, DTS, and SP in the 48, 64, and 128 scenarios

The experimental work shows that designers working with CBA, DTS, GPE and SP must provide an initial floorplan to reach feasible optimized solutions. This limitation makes these representations unsuitable for architectural exploration as it is not always immediate to obtain a feasible configuration. Moreover, the selected seed might bias the search process, exploring only a limited region of the solution space. Thus, there is a **need for techniques that do not require any initial knowledge** of the configuration of the target architecture.

The multi-objective GPE version does not produce optimal floorplans since the configurations obtained with CBA, DTS, GPEf, and SP exhibit a shorter wire length and a lower temperature. However, the solutions obtained with GPE (with NSGA-II) indicate the existence of a **tradeoff between temperature and wire length**. It is therefore necessary to provide a wide range of optimal solutions so that chip manufacturers can select the most suitable floorplan according to their own criteria. To approximate as much as possible the optimal set of solutions that simultaneously minimize wire length and temperature, **multi-objective techniques are necessary**.

In this work, we use multi-objective heuristics to obtain thermally and performance optimized configurations of large 3D MPSoCs. In particular, Multi-Objective Evolutionary Algorithms are specially suitable for our purpose since these population-based heuristics approximate in a single run the set of optimal solutions that simultaneously minimize the targeted conflictive objectives. Moreover, these techniques are compatible with several parallel implementations allowing for faster architectural exploration tasks.

Chapter 4

Heuristics for Multi-objective Optimization

Heuristics are widely employed to solve complex problems when exact or exhaustive approaches are impractical. Many of the demonstrated NP-problems have a direct application in everyday life such as the Traveling Salesman Problem, Staff Scheduling, Production Planning or Job Sequencing among many others. As of today, it is believed that these problems require at least superpolynomial algorithms, even though it is not discarded that faster algorithms can be found. Alternative options exhibiting a lower computational complexity are therefore necessary. Typically, heuristic methods are adopted to find good enough solutions in a short time. Note that the specific constraints of a given context will determine the management of the tradeoff between runtime and quality of the solutions. For instance, in a dynamic context, it might be necessary to obtain good solutions in a really short time. On the other hand, runtime is not always a hard constraint in design time in which the optimality of the solutions is mandatory.

A number of heuristics have been proposed for problem solving that can be classified according to whether a single (Tabu Search, Simulated Annealing) or a group of candidate solutions are managed (Evolutionary Algorithms, Swarm Intelligence). We now give a brief overview of the most common heuristics employed for problem solving:

1) Tabu Search: Presented in [62], this method explores the solution space with

local search techniques combined with the management of a list of forbidden (tabu) neighbours.

2) Simulated Annealing: Proposed in [81] and [150], SA reproduces the slow cooling of metals. In the search procedure, this process is translated into a slow decrease of the acceptance probability of worse solutions.

3) Evolutionary Algorithms: These algorithms consider a set of solutions that are iteratively evolved following a process inspired in Darwin's concept of evolution.

4) Swarm Intelligence: These decentralized and self-organized methods consider a number of independent agents that interact with each other and with the environment according to local rules. The term was introduced by Gerardo Beni and Jing Wang [14] and has inspired a number of proposals such as Ant Colony Optimization [47] or Particle Swarm Optimization [80] among others.

In Chapter 3, the need for multi-objective approaches to the thermal-aware floorplanning of large 3D MPSoCs has been demonstrated. Note that multi-objective optimization refers to optimization problems in which at least two conflictive objectives are targeted. As explained in the referred chapter, both SA and EAs are predominant in previous proposals. Regarding the first, the guiding function of SA engines in multi-objective contexts is typically a weighted sum of the targeted objectives, thus reducing several objectives to a unique function. This technique is extended in some proposals such as [12] where a Pareto Archive is managed to determine the acceptance probability of new solutions. However, Multi-objective Evolutionary Algorithms (MOEAs) are generally used to tackle multi-objective problems. These algorithms extend Evolutionary Algorithms with techniques and strategies that introduce the concept of Pareto optimality. For the fully understanding of the search process involved in MOEAs, it is necessary to introduce EAs properly.

4.1 Evolutionary Algorithms

EAs are a subset of the so-called Evolutionary Computation (which in turn falls within the field of Artificial Intelligence), mainly employed for problem solving and machine learning. Evolutionary Algorithms have been successfully employed in a variety of fields such as design, control, robotics, pattern recognition, scheduling, and classifier systems among many others (see [107] and therein). Moreover, these algorithms have been tested and shown to be robust with a variety of problems presenting different characteristics such as continuous or discrete variables [48] [152], low or high dynamism [112], multimodal and deceptive [127] [5], low or high epistasis [89], etc.

4.1.1 History

Evolutionary computation is inspired in the concept of biological evolution proposed by Darwin in *"On the origin of species by means of natural selection"* [43]. Darwin postulated that evolution was guided by natural selection. In a given environment, the fittest individuals have a higher survival probability and thus, a higher chance of transmitting their genetic information to future generations. As was shown later, random mutations modify the genotype of a given individual, resulting in a modified fitness, i.e. a different survival probability. Thus, environmental changes are beneficial for the species better adapted to the new scenario (climate changes etc...).

These evolutionary concepts can be adapted, recreated and simulated with the aim of solving problems. In this case, **candidate solutions evolve throughout the optimization process**. The analogy of biological evolution has been employed in several optimization approaches that present different procedures such as genetic operators or replacement policies among others. However, these proposals present some common features and patterns in the flow of the optimization process. Arguably one of the main features of these bioinspired heuristics is the management of a population, allowing to simultaneously search in different regions of the solution space. Note that, the employed terminology follows the analogy of natural evolution. Thus,

- Iterations are called *generations*.
- Candidate solutions are called *individuals*.
- The set of individuals managed by the algorithm receives the name *population*.
- In each generation, a new set of individuals is obtained from the current population. New individuals are referred to as *offspring*.

Depending on the concrete algorithm, genetic information can be exchanged through *crossover*, and changes can occur due to random *mutations*. Evolutionary algorithms can be classified into the following three main categories:

1 Genetic Algorithms (GA): introduced by John Holland in the 1960s [65], these algorithms consider a population of *chromosomes* coded as bit strings that represent candidate solutions to a given problem. These chromosomes are then iteratively evolved with the help of several genetic operators. A representative characteristic is the implementation of a crossover operator that combines the genetic information of different candidate solutions. Further details of this strategy are given in the next section.

2 Evolution Strategies (ES): Ingo Rechenberg and Hans-Paul Schwefel initially proposed a loop-based optimization $(1+1)$-Evolution Strategy in which a parent gives birth to a mutant solution [125, 133]. Then, the best of the two considered individuals is selected to be the parent of the next generation. This approach has been modified to obtain several mutants out of a single parent, called $(1+\lambda)-ES$, or to consider several parents and mutants in each generation $(\mu,\lambda)-ES$.

3 Genetic Programming (GP): This category is considered an extension of GAs in which computer programs are evolved to perform a given computational task. Their invention can be attributed to Lawrence J. Fogel (see [54] and therein) who approached the discovery of finite-state automata with evolutionary algorithms. However, this area of research was brought into focus by relevant contributions proposed by Nichael L. Cramer [35] and John R. Koza [87].

4.1.2 EAs Typical Flow

In a nutshell, initial solutions to a given problem are created in an artificial environment in which selection takes place according to a designed fitness function. Note that the initial individuals can be provided or a random population can be created. These solutions are then evolved by means of selection, crossover, and mutation operators until a stop criterion is met. Next, we describe the different steps involved in the optimization process as shown in Figure 4.1. Note that, as mentioned above, some proposals implement only a subset of these steps.

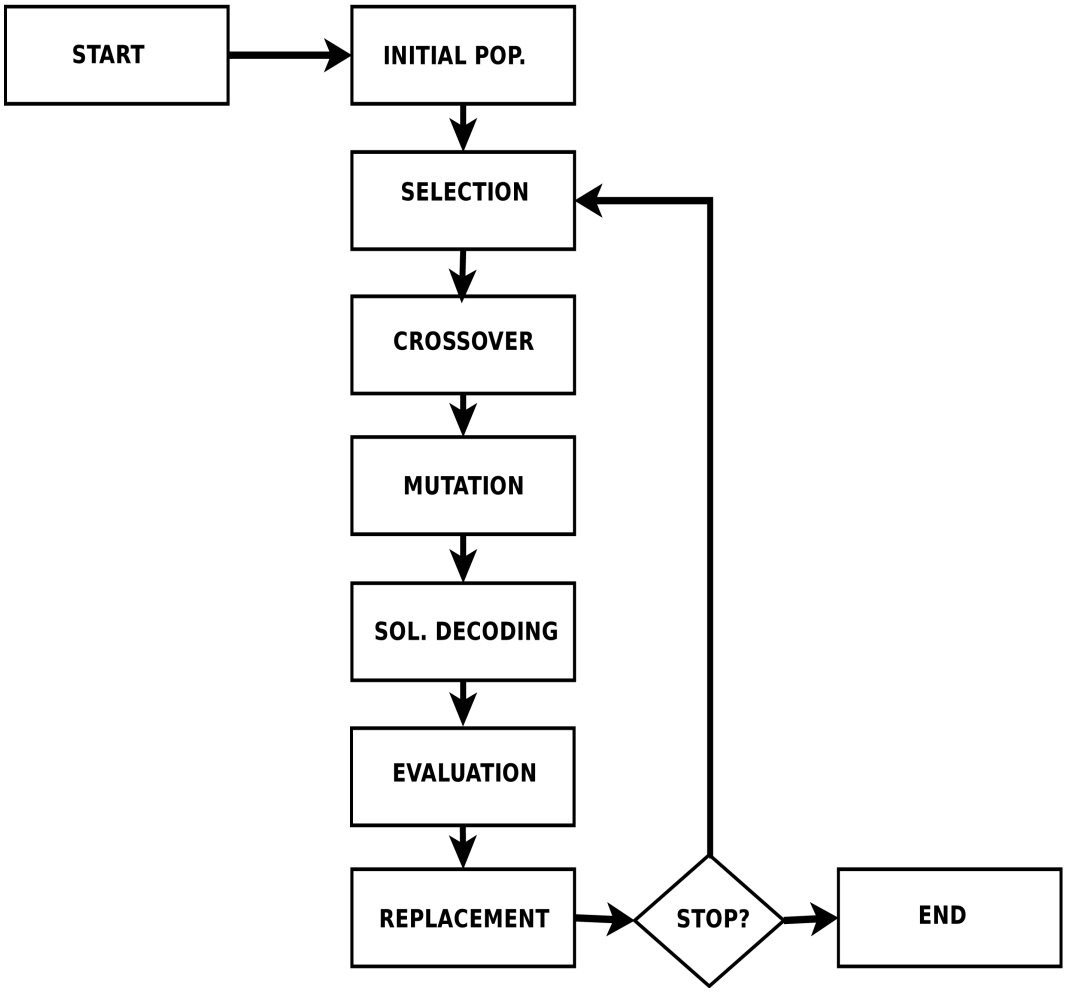


Figure 4.1: EA typical flow

Solution Coding

To apply the depicted scheme, initial solutions must be created and appropriately coded. The genetic operators explained later are applied to the coded solutions, i.e. to their *genotype*. The representation must be carefully designed as it directly impacts the cardinality of the solution space, as well as the fitness landscape of a given problem.

Selection

The Selection operator determines which individuals will be used to generate the offspring. Generally, the better the fitness of an individual, the higher the probability of being selected is. However, individuals exhibiting a low fitness value are typically allowed to transmit their genetic information to the next generation with a reduced probability. It is due to the fact that, in some scenarios, solutions presenting a low fitness might exhibit a genotype similar to an optimal solution.

Crossover

The exchange of genetic information between individuals is performed through a crossover operator. The genotypes of the mated individuals are recombined to create one or two new individuals. The employed representation must be taken into account to select or design an appropriate crossover operator as it impacts the exploration carried out during the search process. To implement an appropriate recombination of the individuals, both representation and problem characteristics must be considered.

Mutation

The mutation operator performs random and infrequent changes in the genotype of individuals. These changes can impact the fitness of these individuals, resulting in a competitive advantage or in a drawback. As in natural evolution, mutations can also be neutral, meaning that the survival probability of the modified individual remains unchanged. If a mutation deeply modifies the genetic information of a given individual, then the search process will explore a different region of the solution space. On the other hand, when a mutation introduces only a slight change, the exploitation of a given region of the solution space is performed. Therefore, as for

the crossover operator, the mutation operator must be representation-dependent and has a great impact on the search dynamics of the designed algorithm.

Solution Decoding

Coded solutions often need to follow a decoding process to be evaluated. Thus, the decoding of the solutions can be seen as a function that produces a *phenotype* from a genotype. Depending on the representation, this process might exhibit a high computational complexity, producing a bottleneck in the optimization process. Again, the design of the representation is important to avoid the potential overhead.

Evaluation

A fitness function is employed to evaluate the different individuals. The fitness value assigned to each individual impacts on the probability of survival, as well as on the chances to transmit its genetic information to the following generation. An inexact design of the fitness function can impede convergence or lead to a suboptimal solution. Fitness functions often are the bottleneck of these algorithms. Therefore, a low computational complexity is desirable. A fitness approximation strategy can be employed when the fitness evaluation exhibits a high computational cost.

Replacement Policies

Once the generated offspring has been evaluated, a replacement strategy selects the individuals that will survive and form the population of the next generation. Therefore, the replacement policy impacts the selection pressure of the evolutionary process, which in turn has an effect on the convergence of the managed population. There are mainly two replacement schemes employed in the field of evolutionary computation:

1 Generational: the created offspring entirely replaces the old population

2 Steady-state: the old population and the generated offspring compete to survive.

Elitism, i.e. forcing the survival of fittest individuals, is necessary. Otherwise, the genetic information obtained in previous generations can be lost, resulting in a random search of the solution space.

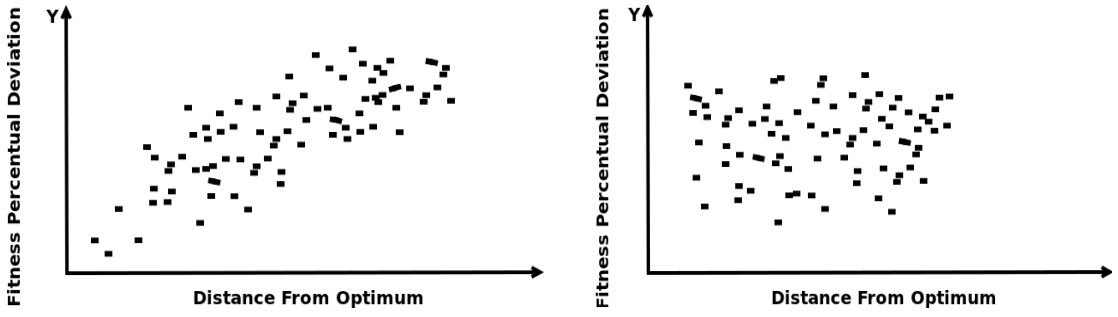
4.1.3 Tuning of EAs: Exploration vs. Exploitation

Some of the previous procedures present a stochastic behavior that can be controlled trough hard coded values such as the mutation rate or the crossover probability. A suitable adjustment of these parameters is crucial for an appropriate operation of the algorithm. In fact, genetic operators must be designed to obtain a suitable balance between exploration and exploitation. One of the indicators used to trace this balance is the diversity of the population. If exploitation prevails over exploration, the algorithm will exhibit a faster convergence, translated into a faster runtime. However, the search process is likely to get stuck in local optima. On the other hand, if the algorithm is not capable of exploiting promising regions of the solution space, the runtime might be increased and it is also probable to reach suboptimal solutions. Note that, depending on the specific requirements, a local optimum might be an acceptable solution if obtained in a reduced convergence time.

Ideally, the search should cover all the interesting regions of the solution space while focusing on promising regions to gradually improve the discovered solutions. Following this idea, Moscato [114] proposed the inclusion of local search operators in the optimization loop giving birth to Memetic Algorithms (see [88] and therein). The name is inspired in Richard Dawkins concept of *meme*, defined as a unit of cultural information [45]. Local search operators are a tool to perform exploitation of the solutions with a locality basis. To this end, a neighborhood relationship has to be established between the different candidate solutions by means of an information-based distance metric. Note that this distance measure is obtained from the comparison of the genotypes of the different solutions rather than with their fitness values. The study of the correlation between the genetic information of the candidate solutions and their fitness receives the name of Fitness Landscape Analysis. This analysis is specially relevant in the context of heuristic optimization as it can provide a deep insight on the distribution of local and global optima of a given problem, and on the suitability of the search dynamics of a given heuristic. The retrieved information can be then be used to guide the design of suitable EAs [74].

4.1.4 Fitness Landscape Analysis

It has been shown that different landscapes exhibit a different difficulty for heuristic problem solving. In fact, problems can be classified according to several measures retrieved from this analysis such as the distribution of local and global optima, multimodality (two or more global optima), deception (global optimum cannot be reached from local optima), or ruggedness (many local optima surrounded by deep valleys) of the fitness landscape [153]. In the context of heuristic optimization, the "hardness" of a problem is not necessarily related to its computational complexity. Thus, problems with the same computational complexity are not necessarily equally hard for heuristic solving, and high computational complexity problems might be relatively easy to solve with heuristics. Moreover, instances of the same problem might present very different fitness landscapes, resulting in very different difficulties [154]. To illustrate this concept, we present below a typical analysis used to assess these properties, namely the fitness-distance correlation. Figure 4.2 schematically shows the analysis of the distance of the solutions to the global optimum and their fitness in two different scenarios. The first problem (Figure 4.2(a)) can be considered easy to solve with heuristics as a clear correlation between fitness and distance can be detected. In such scenario, the closer the solutions are to the global optimum, the higher their fitness value is. Therefore, a simple heuristic iteratively looking for the best neighbour would easily reach the optimum solution of the problem. On the other hand, the case depicted in Figure 4.2(b) does not show any fitness-distance correlation. Therefore, this problem is likely harder to be solved with heuristic approaches.



(a) Easy Scenario (b) Hard Scenario

Figure 4.2: Fitness-Distance Correlation Analysis

4.2 Multi-Objective Evolutionary Algorithms

In this section, we introduce Multi-Objective Evolutionary Algorithms (MOEAs). These algorithms incorporate the Pareto optimality notion, as opposed to mono-objective EAs in which linear aggregative functions are used to tackle multi-objective optimization problems. Some multi-objective optimization concepts must be introduced first to explain the state-of-the-art MOEAs and discuss their suitability for the thermal-aware floorplanning problem.

4.2.1 Introduction to Multi-Objective Optimization

It can be said that real-world problems are typically multi-objective because several goals such as benefits, cost, quality of service, efficiency, time etc. are usually targeted. For instance, a distribution itinerary planning equivalent to a Traveling Salesman Problem typically requires optimal solutions minimizing the length of the path, but also the cost, and the necessary time to perform the whole route. In fact, cost and time might have an impact on the final benefits associated to the distribution or on the resulting quality of service.

Initial proposals combined these different objectives into a single objective function, reducing the search to a mono-objective problem looking for a single optimal point. However, "true" multi-objective approaches consider a fitness value for each of the targeted objectives as, generally, there does not exist a single point that simultaneously minimizes all the objective functions. Thus, the goal is to obtain a family of optimal vectors, called the Pareto optimal set or front. The construction of such set is based on the concept of Pareto optimality introduced by Francis Ysidro Edgeworth [50] and generalized by Vilfredo Pareto [122]. The vectors forming the Pareto optimal set receive the name of nondominated. Let is consider a minimization problem targeting n objectives, then

Definition 1. *A vector* $u = (u_1, u_2, ..., u_n)$ *is dominated by* $v = (v_1, v_2, ..., v_n)$ *if and only if* $\forall i = 1, ..., n, v_i \leq u_i \wedge \exists j = 1, ..., n : v_j < u_j$

However, in real-world problems in which exhaustive approaches are impractical, it is not possible to assert that a given solution is not dominated by any other solution. Thus, the goal of multi-objective optimization is not necessarily to obtain the Pareto Optimal front but rather to approximate it as much as possible. Figure 4.3 illustrates this idea. An approximated Pareto front (squares) found by a given heuristic is depicted. All the solutions of this front are the nondominated solutions found in a run of the considered algorithm. On the other hand, the triangles represent dominated solutions. As explained later, the Pareto dominance concept is typically used as a criterion to rank and select the fittest solutions.

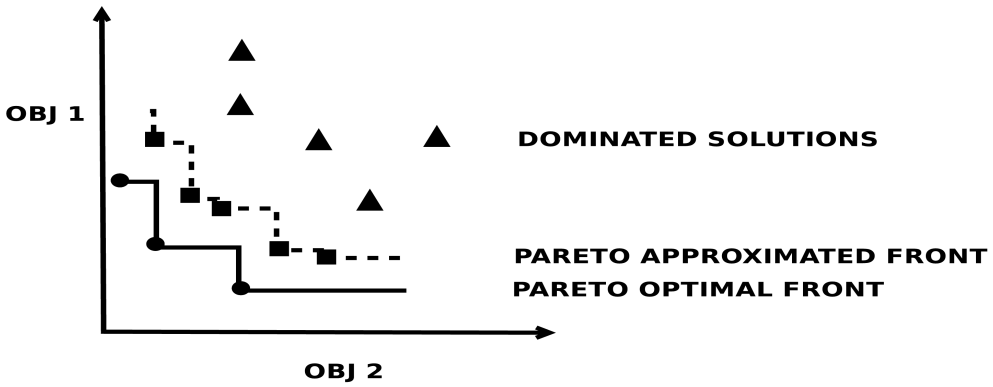


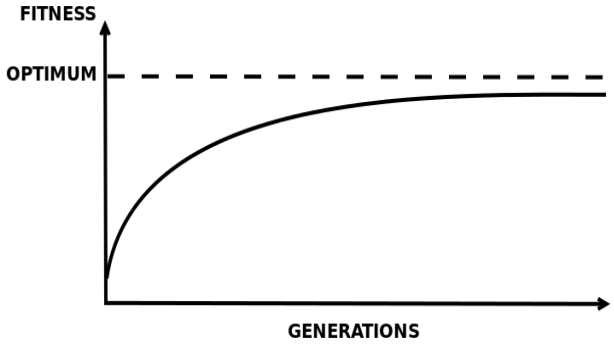
Figure 4.3: Example of an approximated Pareto front

4.2.2 EAs + Pareto Optimality

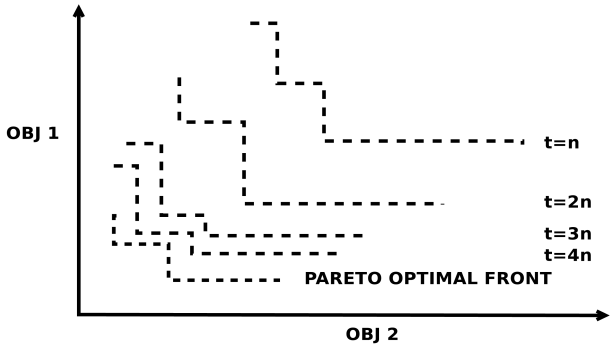
MOEAs extend Evolutionary Algorithms with techniques and strategies that introduce the concept of Pareto optimality to guide the managed population towards the Pareto optimal set. The main advantage of evolutionary algorithms, when applied to solve multi-objective optimization problems, is the fact that they typically optimize sets of solutions. This allows to compute an approximation of the entire Pareto optimal front in a single algorithm run.

We now propose a schematic example to clarify the difference between approaches in which linear aggregative functions are used to tackle multi-objective optimization problems and proposals that include the notion of Pareto optimality. In the first

case, as depicted in Figure 4.4, fitness evolution approximately depicts a function asymptotic to a horizontal line corresponding to the best obtained fitness (Figure 4.4(a)). On the other hand, in a MOEA context, the managed population should iteratively approximate the Pareto optimal set as illustrated in Figure 4.4(b).



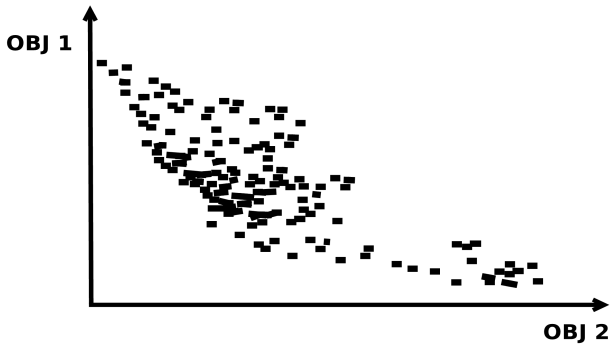
(a) Mono-objective Scenario



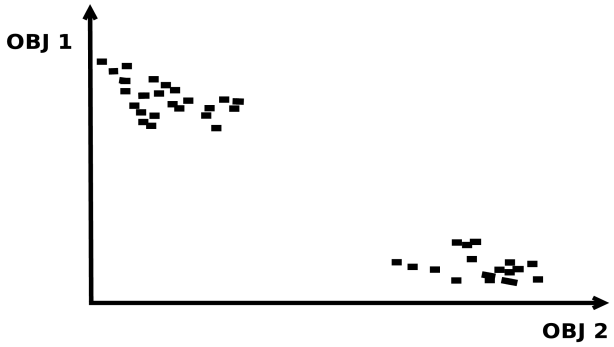
(b) Multi-objective Scenario

Figure 4.4: Convergence analysis in Mono-objective and Multi-objective scenarios

MOEAs present different strategies based on the Pareto dominance concept to rank and select the best fitted individuals. The first MOEA is attributed to Schaffer [132], who proposed the Vector Evaluation Genetic Algorithm (VEGA) in 1984. In this proposal, selection is performed considering separately the different objectives leading to an undesirable speciation phenomenon. In fact, solutions presenting high fitness values in only one of the targeted objectives prevail, while solutions presenting a good overall tradeoff tend to be discarded. As a consequence, the obtained solutions approximate only partially the Pareto optimal set. The VEGA selection strategy is schematically illustrated in Figure 4.5.



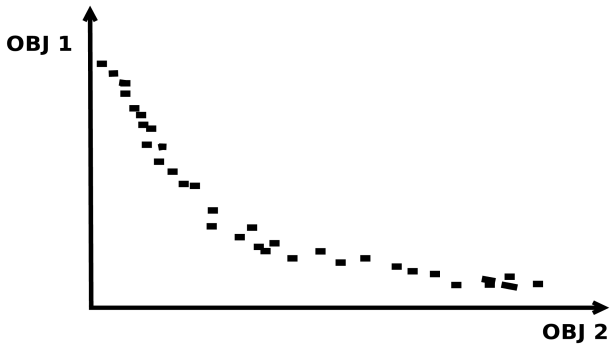
(a)



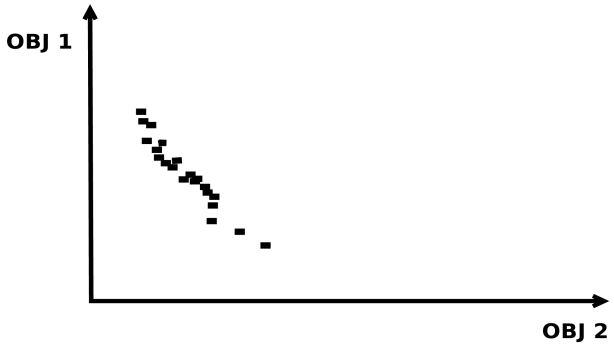
(b)

Figure 4.5: A set of points are depicted in Figure 4.5(a) according to their fitness in the two targeted objectives. The points selected according to the VEGA strategy are shown in Figure 4.5(b)

MOEAs did not receive much attention until the mid-1990s when the subject became a relevant research topic. Later MOEA proposals are classified in two generations according to Coello [26]. The first is composed of proposals introducing a hierarchy based on Pareto dominance and niches to maintain the diversity of the population and avoid premature convergence issues. This idea is illustrated in Figure 4.6, in which the example of Figure 4.5 is extended. As can be seen, a selection strategy based on the the concept of niche allows to completely cover the Pareto front of nondominated solutions (Figure 4.6(a)) while inappropriate strategies lead to focus on the exploitation of a region of the solution space as depicted in Figure 4.6(b). The most representative proposals of the first generation are Nondominated Sorting Genetic Algorithm (NSGA) [140], Niched-Pareto Genetic Algorithm (NPGA) [66], and Multi-Objective Genetic Algorithm (MOGA) [55].



(a)



(b)

Figure 4.6: A set of points selected according to a niche strategy is depicted in Figure 4.6(a). Figure 4.6(b) shows a scenario in which the selected points are concentrated in a given region of the solution space.

The algorithms that belong to the second generation incorporate elitism using selection and a secondary population. This secondary population is often maintained as an external Pareto archive containing the nondominated solutions found during the search process. Some of the proposals classified in this category are Strength Pareto Evolutionary Algorithm (SPEA) [162], Strength Pareto Evolutionary Algorithm 2 (SPEA2) [161], Pareto Archived Evolution Strategy (PAES) [83], Niched Pareto Genetic Algorithm 2 (NPGA 2) [53], Pareto Envelope-based Selection Algorithm (PESA) [32], and Nondominated Sorting Genetic Algorithm II (NSGA-II) [46] in the following section.

We focus on the latter, NSGA-II, as it is the algorithm employed later in this work. In fact, this method together with SPEA-2, has become a standard approach to deal with multi-objective optimization problems. We present an overview of NSGA-II.

4.2.3 NSGA-II

As in its predecessor version NSGA, the main feature of this method is the selection of the best fitted individuals according to a ranking established on the basis of Pareto dominance. Thus, a given individual receives a ranking value according to the number of individuals that dominate it. Several layers of candidate solutions corresponding to Pareto fronts are then created, and the individuals forming the best layers are selected for the next generation. If only a few individuals of a given layer must be selected, a crowded comparison is employed. This strategy does not need any additional parameter, as opposed to fitness sharing methods. Figure 1 shows the pseudocode of the NSGA-II.

Generate the initial population $P(t)$ of μ individuals, each represented as a set of real vectors, (x_i,η_i), $i = 1, ..., \mu$. Both x_i and η_i contain N independent variables: $x_i = \{x_i(1), ..., x_i(N)\}$, $\eta_i = \{\eta_i(1), ..., \eta_i(N)\}$;
Evaluate the objective vectors of all individuals in $P(t)$;
Calculate the rankings and crowding distances of all individuals;
Execute $DominanceChecking(P(t), C, S)$;
Execute $NonDominatedSelection(P(t), C, S, V, \mu)$;
for $g \leftarrow 1$ **to** $NumGenerations$ **do**
    for $i \leftarrow 1$ **to** $\mu/2$ **do**
        select two parents $Parent_1$ and $Parent_2$ from $P(t)$ using the tournament selection method;
        recombine $Parent_1$ and $Parent_2$ using the crossover operator to produce two offspring stored in the temporary population P^2;
    end
    The population P^2 contains μ individuals;
    Mutate individuals in P^2 to generate modified individuals stored in the temporary population P^3;
    Evaluate the objective vectors of all individuals in P^3;
    Combine the parent population $P(t)$ with P^3 to generate a population P^4 containing 2μ individuals;
    Check the dominance of all individuals in P^4 by executing $DominanceChecking(P^4, C, S)$;
    Select μ individuals from P^4 and store them in the next population $P(t+1)$. The individuals are selected by executing $NonDominatedSelection(P^4, C, S, V, \mu)$;
end
Return the nondominated individuals in the last population;

Algorithm 1: NSGA2 flow

4.2.4 Tuning of MOEAs

As explained in Section 4.1.3, EAs must be tuned to operate adequately. In the case of MOEAs, a number of strategies introducing Pareto optimality have been proposed. These techniques allow to look for nondominated solutions in different regions of the solution space in a simultaneous manner. However, convergence towards the optimal Pareto Set can present several problems, such as setting inappropriate weights to the different objectives. As a consequence, the search might be focused on the optimization of a subset of the objectives. Figure 4.7 illustrates this idea; it can be seen that the obtained solutions only approximate the Pareto optimal set partially (obtaining good fitness values only for the first objective).

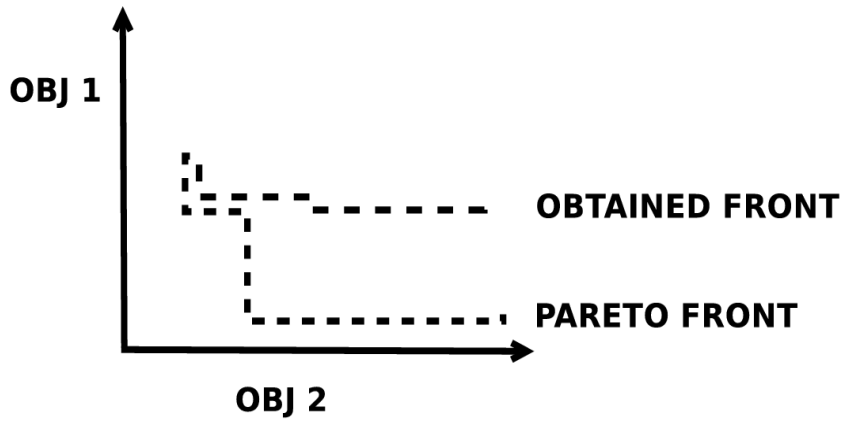


Figure 4.7: Example of undesired result in a Multi-objective scenario

The search behavior depicted in the figure typically takes place when dealing with multiple (more than 2) objectives. In fact, several targeted objectives might happen to be equivalent, resulting in an unbalanced setting of the weights of the different objectives. A potential solution to the referred problem consists in eliminating redundant objectives. Moreover, dimensionality reduction is a crucial step in many-objective optimization problems, as it is commonly accepted (although there is no formal proof) that MOEAs perform poorly with a high number of objectives. This behavior is due to the fact that most of the individuals tend to become nondominated, resulting in a loss of selective pressure [136].

The constraints affecting the feasibility of the candidate solutions and the ranges

of acceptability of the targeted objectives represent another difficulty for multi-objective optimization. For instance, in some scenarios such as the thermal-aware floorplanning, only a range of values (temperatures) result in feasible chips. However, considering a predetermined range of values might negatively impact the search process, reaching only a subset of the solution space and impeding convergence towards the whole optimal Pareto set.

In multi-objective contexts, the bottleneck caused by the evaluation step is exacerbated as a fitness value is computed for each of the targeted objectives. Therefore, parallel approaches are often necessary to reduce the required optimization time.

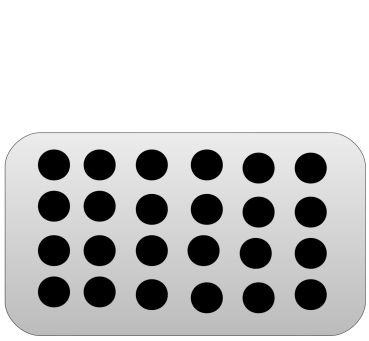
4.3 Parallel Evolutionary Algorithms

Evolutionary Algorithms present an inherent parallelism that can be exploited to obtain faster runtimes. There are several implementation possibilities that typically introduce changes in the design of these algorithms, namely in the structure of the population. These structured populations were engineered to match the architecture of the hardware platforms in which they were run. Figure 4.8 schematically depicts, from coarser to finer, the main design alternatives [3]. It is worth noting that the following proposals were conceived to deal with mono-objective problems. However, Parallel Evolutionary Algorithms (PEAs) can be adapted and are equally suitable for multi-objective contexts.

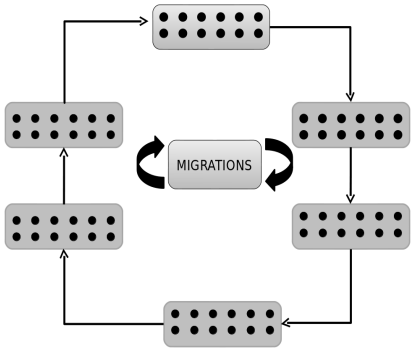
4.3.1 Panmictic Population

This approach, depicted in Figure 4.8(a) corresponds to the classic EA in which all the individuals form a single population. Thus, any two individuals can be selected as parents to mate and obtain the offspring. In a similar way, all the individuals compete against each other to survive and can be replaced by new individuals.

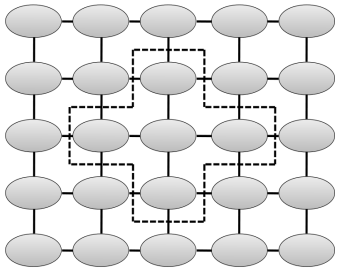
Following this approach, it is typical to parallelize the evaluation step of the algorithm as it usually represents the bottleneck of the optimization process. To this end, the parallelization can be implemented using a Master-Worker Model suitable for multithreaded architectures in which each worker evaluates a subset of the



(a) Panmictic Population



(b) Island Model



(c) Cellular Model

Figure 4.8: Evolutionary Algorithms with structured populations

population. In such case, the expected speedup would be approximately linear with the number of threads.

There are also General Purpose Graphic Processor Unit (GPGPU) approaches that aim to reduce the runtime of the evaluation function, whether or not several individuals are evaluated concurrently [105].

In this thesis, we will present both a multi-threaded implementation of the Master-Worker model and a GPGPU evaluation of the population to speedup the optimization of large 3D MPSoCs.

4.3.2 Island Model

Figure 4.8(b) shows a PEA implementing a decentralized Island Model in which individuals are distributed among the different islands. Evolution takes place independently in these islands with the exception of migrations. In fact, individuals are migrated from an island to another to spread the information obtained in a specific island throughout the system. Such individuals are referred to as *migrants*. The migration policy determines the way migrants are selected and how replacement takes place in the receiving islands. The network structure governing these transactions receives the name of *topology*.

This approach is well-adapted for multiprocessors as a given island can be mapped to a single processor. The island model is also typically employed in clusters, in which the elevated number of computing resources perfectly fits this distributed model.

This model is suitable for GPGPU architectures as well. In fact, PEAs can be designed strategically to maximize speedup considering the specific memory hierarchy restrictions and programing paradigms provided by modern GPUs. For instance, several CUDA-based ([119]) implementations have been proposed matching blocks and threads to islands and individuals respectively [147] and [123].

The island model has been proved to be beneficial not only in terms of runtime, but also in terms of fitness of the obtained solutions. In fact, distributing the search over several nodes allows for a better exploration of the solution space, leading to a superlinear speedup with respect to a panmictic scheme run in a single core processor [20] and [21]. As a result, it is nowadays usual to design PEAs exhibiting a higher degree of parallelism than allowed by the hardware platform in which the algorithm is run. In fact, it is usual to simulate an island model in a single core processor. For instance, it might be beneficial to design a PEA with two 50 individual islands rather than considering a single population with 100 individuals even if the runtime remains unchanged.

The impact of the network topology [3], [152], [48] and the migration policy

[4], [68], [103], [128], [5], [111] has been extensively analyzed. In fact, not all the topologies are suitable for a given problem. Moreover, a recent work shows that the optimal topological characteristics and the problem structure are correlated [8]. The design of the network topology governing migrations is in fact crucial to achieve the necessary balance between exploration and exploitation.

4.3.3 Cellular Evolutionary Algorithms

Cellular Evolutionary Algorithms are decentralized EAs with structured populations in which each individual interacts with its neighbours exclusively. The established locality parameter and topology have a great impact in the search dynamic of the algorithm [60], [59].

This approach is suitable for SIMD architectures, as selection and crossover are implemented with a locality basis and the mutation and evaluation of the different individuals can be performed independently.

In this work, we employ MOEAs to perform the optimization of large 3D MPSoCs. These algorithms represent a suitable tool to perform the thermal-aware floorplanning of the studied architectures since exhaustive approaches are impractical and several objectives must be targeted. The proposed floorplanners are based on NSGA-II and their correct operation has been statiscally validated. Moreover, the impact of using different representations is extensively analized. We also propose parallel approaches resulting in a significantly faster optimization process. In particular, a Master-Worker model and a GPU implementation are proposed to speedup the evaluation phase of the proposed techniques,

4.4 Conclusions

In this chapter, we have introduced the family of Evolutionary Algorithms (EAs), which are population-based heuristics inspired in Darwin's concept of evolution. In turn, Multi-Objective Evolutionary Algorithms (MOEAs) incorporate the concept of Pareto optimality and represent an efficient tool to deal with optimization problems in which two or more conflicting objectives are targeted. Special focus has

been dedicated to a state-of-the-art MOEA, namely Nondominated Sorting Genetic Algorithm II (NSGA-II) as it is employed later in this work. In fact, MOEAs are specially suitable for the thermal-aware floorplanning problem because both temperature and wire length must be minimized. Several parallel EA approaches resulting in a faster exploration of the solution space have also been presented.

In previous chapters, we have introduced the state-of-the-art of 3D Integration and the most relevant floorplanning proposals. We have also described existing thermal models which, together with multi-objective techniques, represent the necessary tools to attack the thermal-aware floorplanning problem. In the following chapters, the main contributions of this thesis will be presented.

Chapter 5

Multi-Objective Techniques for the Thermal-Aware Floorplanning

As explained in Chapter 2, the optimization of 3D Integrated Circuits can be considered at different abstraction levels corresponding to different design possibilities. These approaches differ from each other in the granularity of the units to be stacked. In this work, the floorplanning of large 3D MPSoCs is approached considering entire cores and memories as the low redesign effort (2D designs can be reused) makes this approach plausible in a short-term future. Even if the power and performance of the different components remain unchanged, it allows to reduce the global wiring of the whole chip, resulting in a better overall performance. The comparative analysis presented in Chapter 3 shows that existing floorplanning techniques are not suitable for the thermal-aware floorplanning of the targeted large 3D platforms. In fact, the analyzed techniques require an initial feasible floorplan to reach optimal solutions, which might not be immediate. Moreover, the selected seed might bias the search process, resulting in a limited exploration of the search space. Another major drawback of the compared techniques is the adopted mono-objective approach, which does not allow to deal with the tradeoff between temperature and wire length in an appropriate manner. Floorplanning algorithms must provide a wide range of optimal solutions simultaneously minimizing the two referred conflicting objectives. The necessary notions of multi-objective optimization are given in Chapter 4. Multi-Objective Evolutionary Algorithms are explained in detail as these heuristics are specially suitable for our purpose, i.e. for floorplanning problems in which two

or more objectives must be simultaneously minimized.

Thus, in this work, we use Multi-Objective Evolutionary Algorithms to attack the fixed-outline floorplanning problem considering entire cores and memories. The studied problem is then equivalent to a multi-objective block placement problem in which both thermal and performance constraints must be considered. The block placement problem is formulated as follows:

Block Placement Problem

All the blocks that model the different components of the system must be placed in the 3D stack, which imposes the physical boundaries of maximum length L, width W and height H. Every block i in the model $B_i(i = 1, 2, \ldots, n)$ is characterized by a width w_i, a height h_i and a length l_i. Note that in the studied cases in this work, $h_i = 1$ for all the considered components. We define the vector (x_i, y_i, z_i) as the geometrical location of block B_i, where $0 \leq x_i \leq L - l_i$, $0 \leq y_i \leq W - w_i$, $0 \leq z_i < H$ (see Figure 5.1). We use (x_i, y_i, z_i) to denote the back-left-bottom corner of block B_i while we assume that the coordinate of back-left-bottom corner of the resultant IC is $(0, 0, 0)$. As opposed to traditional floorplanning problems in 2D, the area of the chip is not initially targeted as we consider a fixed die size.

This chapter starts with the description of the so-called Multi-Objective Floorplanning Algorithm (MFA), presented for the first time in [38], as it represents the starting point of this work. Next, the main following contributions are presented:

- The Multi-Objective Floorplanning Algorithm is validated and compared against state-of-the-art floorplanners
- A multi-threaded implementation of the Master-Worker Model is employed to reduce the runtime of the referred floorplanner
- We introduce knowledge by means of a power profiling step previous to the thermal-aware floorplanning process to better guide the optimization
- We compare the performance and thermal optimization achieved introducing

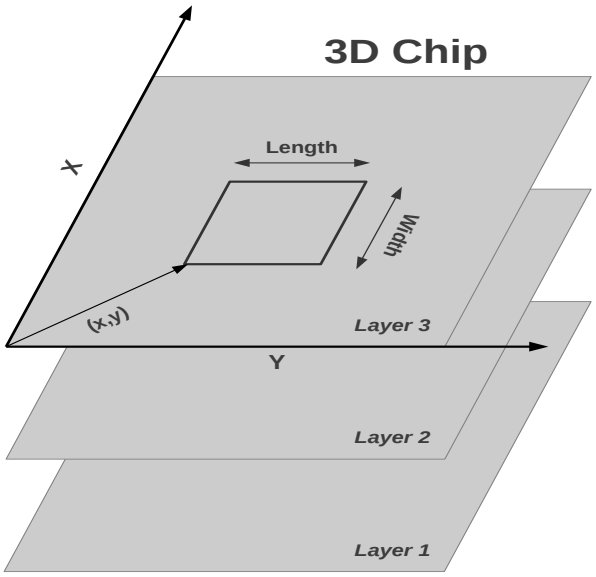


Figure 5.1: Block representation

different thermal models in the optimization loop.

5.1 Multi-Objective Floorplanning Algorithm

In the first part of the section, we describe Multi-Objective Floorplanning Algorithm (MFA) [38], a floorplanner targeting the thermal and performance optimization of large 3D MPSoCs. A validation of the algorithm is proposed in the second part. To this end, the convergence of the algorithm is studied, the proposal is compared against state-of-the-art floorplanners, and the obtained configurations are evaluated in terms of temperature and wire length.

5.1.1 Description of the Algorithm

MFA is a multi-objective evolutionary algorithm based on NSGA-II (see Chapter 4). The floorplanner manages coded solutions that are gradually improved in the evolutionary process to provide thermally and performance optimized configurations of the architecture. MFA can be classified as a hybrid floorplanning approach because the decoding step implements an incremental floorplanning inspired in constructive techniques while the MOEA on top of the decoding procedure is essentially iterative. Figure 5.2 illustrates this idea. The details of the algorithm are given in the following.

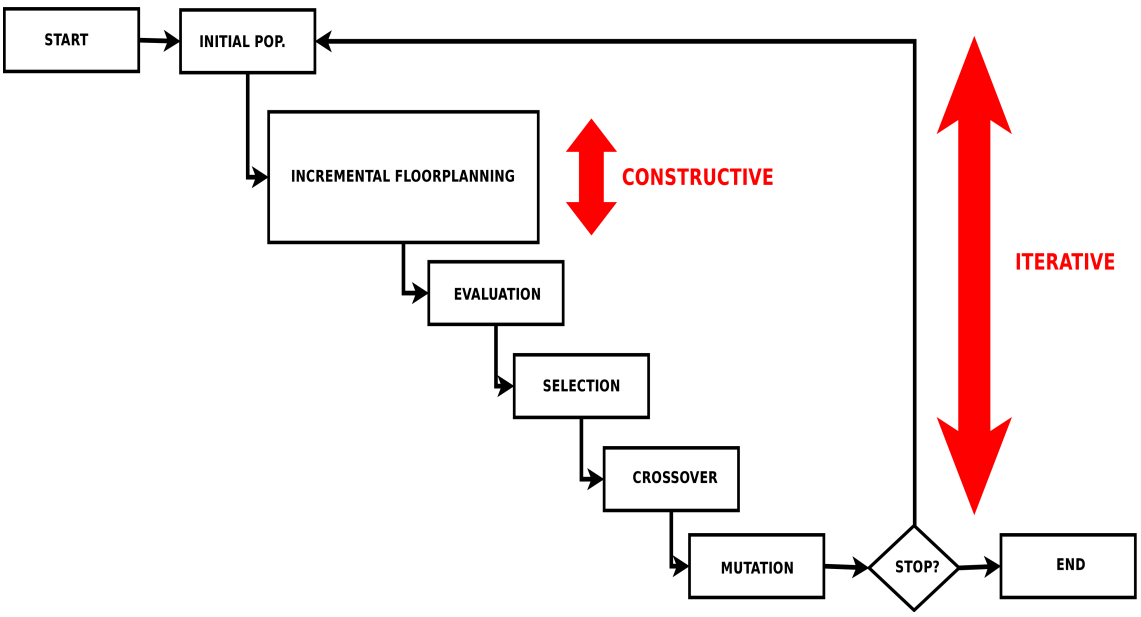


Figure 5.2: MFA: a hybrid floorplanning approach

Representation

A permutation encoding [25] is used in which every chromosome is a string of records representing the different components of the targeted architecture. These records gather information relative to a block, namely label, width, and length. Managing the width and length of the blocks allows to perform rotations, granting further degrees of freedom to the optimization process. Additional characteristics of the blocks such as power densities, connections etc. must be managed by the algorithm. However, this information does not need to be codified in the chromosomes as it is common to all the individuals. Figure 5.3 depicts the representation used in MFA, the example shows a candidate solution (individual) of a platform composed of 6 blocks: 3 processors $C_i(i = 1, 2, 3)$ and 3 memories $L_i(i = 1, 2, 3)$. The order $C_1, L_3, L_1, L_2, C_3, C_2$ determines the placement sequence. Thus, C_1 will be placed first, followed by L_3 and so on.

All the chromosomes must be of size n, where n is the number of blocks to be placed. Therefore, the cardinality of the considered solution space is $n!$. The procedure in charge of the placement of the blocks (decoding of the solutions) which allows to map lists of components into configurations of the architecture is explained later.

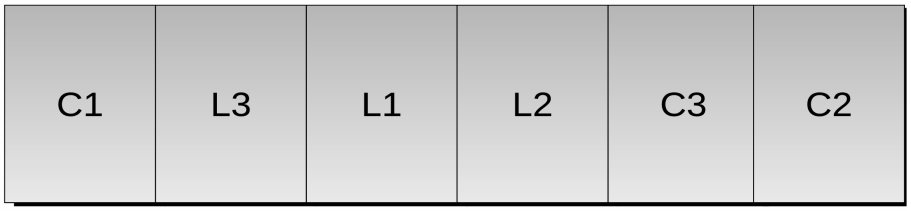


Figure 5.3: Candidate solution coded with the MFA representation

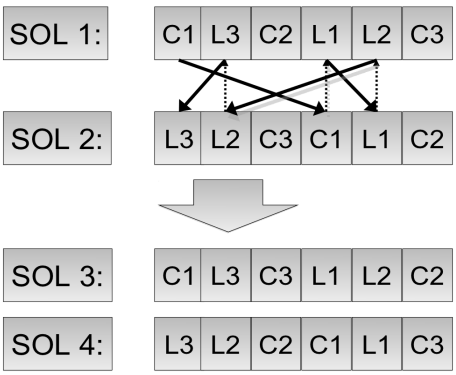
Operators

The operators designed according to the representation are depicted in Figure 5.4 and briefly described below:

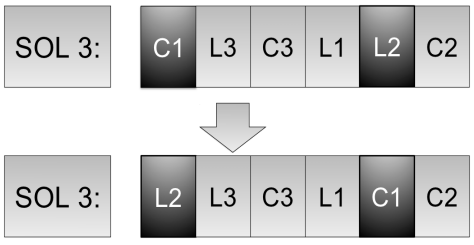
Selection: The selection operator implements a binary tournament strategy. To this end, random couples of individuals are formed and the best solution of each pair is selected.

Crossover: A cycle crossover is used to produce the offspring, this operator must take into account that all the components must appear once and only once in the chromosome (see Fig. 5.4(a)).

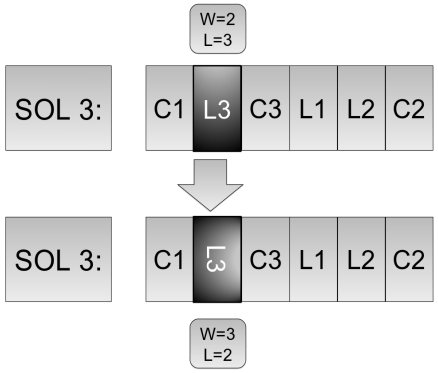
Mutation: The mutation of the solutions is performed in two ways. The first one consists in swapping the position of two blocks in the chromosome, resulting in a change of the placement sequence of the mutated individual (Fig. 5.4(b)). The effect of the second is the rotation of a block (Fig. 5.4(c)).



(a) Cycle crossover



(b) Swap mutation



(c) Rotation

Figure 5.4: MOEA operators: Cycle crossover and two mutation operators (swap or rotate)

Solution Decoding

A sophisticated heuristic is in charge of the placement of the different elements of the architecture (decoding of the solutions). The heuristic performs an incremental floorplanning in which the components are sequentially placed in the 3D stack following the order implied by the solution encoding. In fact, as the exhaustive heuristic alone is capable of obtaining well performing solutions, the MOEA is designed to obtain the optimal order of the components given to the placement heuristic.

Every block B_i is placed considering all the topological constraints, the total wire length, and the maximum temperature of the chip with respect to all the previously placed blocks $B_j : j < i$. The best location for each block is selected depending on whether the block is a relative heat sink or a heat source. If it is a heat source (like a core, for example) the best point is the one with lower temperature to ensure an even thermal distribution. If the block is a heat sink (like a memory) the best point is the one with lower wire length. With this procedure, the authors try to ensure a correct thermal optimization. It is reasonable, since large 3D stacks with more than 48 cores reach prohibitive temperatures (more than 400 K, as can be seen in [38]).

Thus, every block is set in the remaining position (x_i, y_i, z_i) minimizing all the targeted objectives.

Fitness

Once the placement has been performed, the obtained configurations are evaluated according to the three following objectives:

- The number of topological constraints violated (overlapping between different blocks or components out of the borders of the chip).
- The wire length approximated as the Manhattan distance between interconnected blocks.
- The maximum temperature of the chip. The computation of this metric depends on the chosen thermal model.

MOEA Parameters

Regarding MFA parameters, the population size is fixed to 100, and the probabilities of mutation and crossover to the inverse of the number of blocks and 0.9, respectively (as recommended in [46]).

5.1.2 Experimental Validation

MFA presents an exhaustive heuristic in charge of decoding the solutions that is highly time consuming. The computational effort of such heuristic forces to set a low number of generations of the MOEA managing the coded solutions. Therefore, it is necessary to study the convergence of the algorithm to make sure that a reduced number of generations is enough to reach convergence while the managed population exhibits appropriate levels of diversity throughout the search process.

Convergence of the MOEA

As explained in Chapter 4, EAs present an inherent tradeoff between exploration and exploitation that must be managed to ensure the correct behavior of the algorithm. Otherwise the exploration of the solution space will be guided towards a region, avoiding others which might be more promising. Generally, in the initial phase of the evolutionary process, the population presents a high diversity which decreases gradually as the population converges towards a reduced number of solutions. Depending on the problem, it might be desirable to keep high levels of diversity during a shorter or longer period.

The analysis of the convergence is a straightforward method for verifying that premature convergence issues such as getting stuck in local optima are avoided. At the same time, it reveals whether the EA is well engineered or there is room for improvement. A slow convergence with good final results is usually due to a poor representation or not appropriate genetic operators.

Although MFA tackles a three objective problem (feasibility, wire length, and temperature), convergence is studied only in terms of the wire length (W) and thermal response (T) of feasible solutions. Thus, values W and T of feasible solutions are

registered in the bidimensional arrays $W_{r,g}$ and $T_{r,g}$ respectively, where $r = 1 \ldots 30$ represents the run and g the generation ($g = 1 \ldots 250$ in the 48 cores scenario and $g = 1 \ldots 366$ for 128). Then, six matrices are constructed in the following way:

$$\begin{aligned} \mathbf{W_{min}}(r,g) &= \min\{W_{r,g}\}, \\ \mathbf{W_{mean}}(r,g) &= \text{mean}\{W_{r,g}\}, \\ \mathbf{W_{max}}(r,g) &= \max\{W_{r,g}\}. \end{aligned}$$

$$\begin{aligned} \mathbf{T_{min}}(r,g) &= \min\{T_{r,g}\}, \\ \mathbf{T_{mean}}(r,g) &= \text{mean}\{T_{r,g}\}, \\ \mathbf{T_{max}}(r,g) &= \max\{T_{r,g}\}. \end{aligned}$$

This procedure is done for both the 48 and 128 core configurations. Finally, all six matrices are scaled between the minimum and maximum wire length and thermal response respectively, and plotted as shown in Figure 5.5. Note that the convergence in the 64 cores scenario is not studied since we consider that the analysis of architectures composed of 48 and 128 cores is enough to generalize the behavior of the algorithm.

The left-most pictures, corresponding to the minimum values in each generation and optimization run of W and T, show a decreasing behavior eventually reaching the global minimum in almost all optimization runs for 48 cores and, at much slower rate, for 128 cores. The middle pictures show convergence of the mean values of W and T. Their trend is decreasing, starting at 1/2 of the upper bound down to 1/4 in the best cases and 1/3 in the worst. Finally, the convergence of maximum values of W and T is shown in the right-most pictures. Values decrease down to 1/3 of the upper bound in most of the optimization runs.

To the light of these results it is clear that candidate solutions are better fitted as evolution advances. Moreover, since the mean of the last generation is quite close to the maximum in both objectives, less fitted individuals still have a considerable probability of being chosen, attesting that diversity is maintained.

On the other hand, the slow convergence of the mean values, especially in the

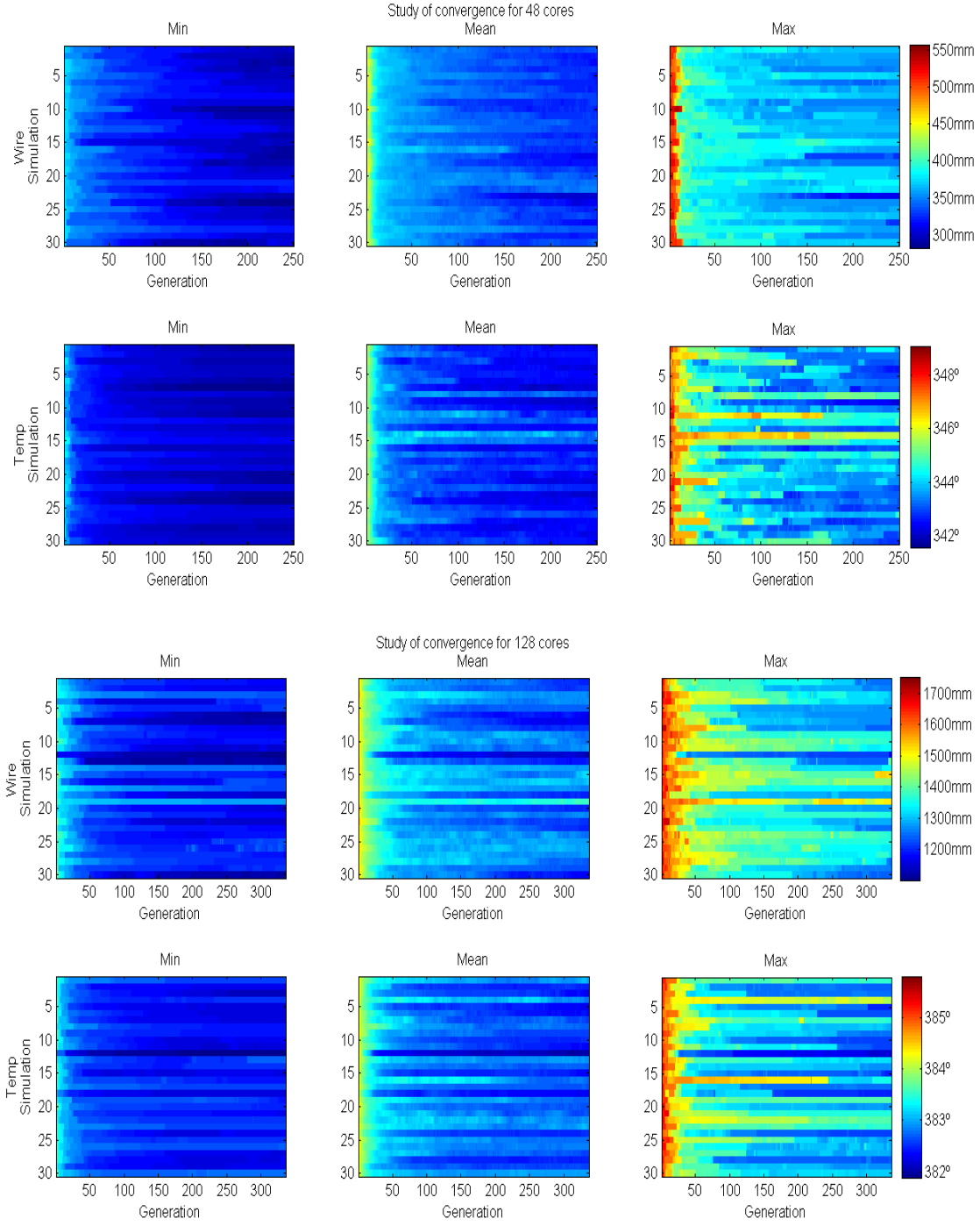


Figure 5.5: Convergence evolution of the minimum, mean and maximum values for objectives W (wire length) and T (thermal response), considering only feasible individuals; for 48 cores (above) and 128 cores (below).

larger problem of 128 cores, indicates that the representation could be better engineered (this goal will be pursued along this thesis).

Comparison Against State-of-the-art Floorplanners

We now extend the comparative study of state-of-the-art floorplanners presented in Section 3.2 with the results obtained with MFA using the same time deadlines. The results are shown in Figure 5.6.

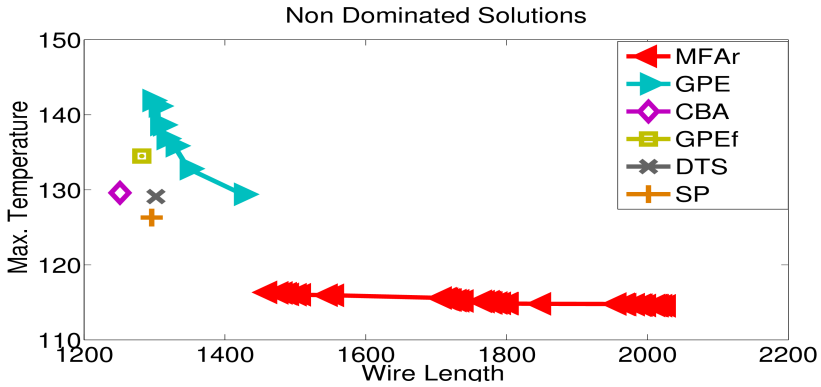
It can be seen that MFA presents a singular behavior and that no domination relationship is established with the other proposals. In fact, MFA explores a different region of the solution space as compared to the rest of representations. Moreover, the difference increases with the number of cores, i.e. when the thermal impact of the architecture is higher. It is due to two facts:

1. MFA does not start with a fixed seed, namely the original Niagara platforms
2. The heuristic employed in MFA to determine the position of a given block depends on whether the component is a core ($\hat{T}$ is minimized) or a memory (W is minimized). Thus, the design of MFA might be leading to reach better temperatures sacrificing wire length.

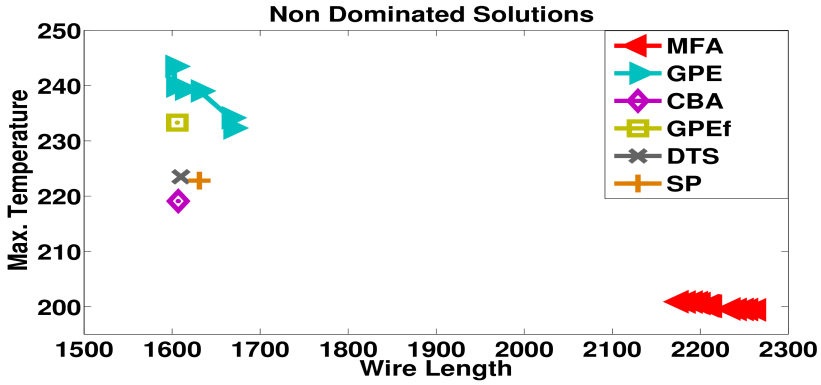
Regarding the second point, in [38] it was shown that the algorithm finds floorplans exhibiting a low enough temperature even when the number of cores increases.

CBA, SP, DTS, and GPE representations do not achieve an appropriate thermal optimization, mainly because they were initially developed to reduce area or wire length. In the thermal aware floorplanning problem, hottest elements must be placed as far as possible in the 3D stack. These representations however place hot elements (cores) right next to each other producing hotspots.

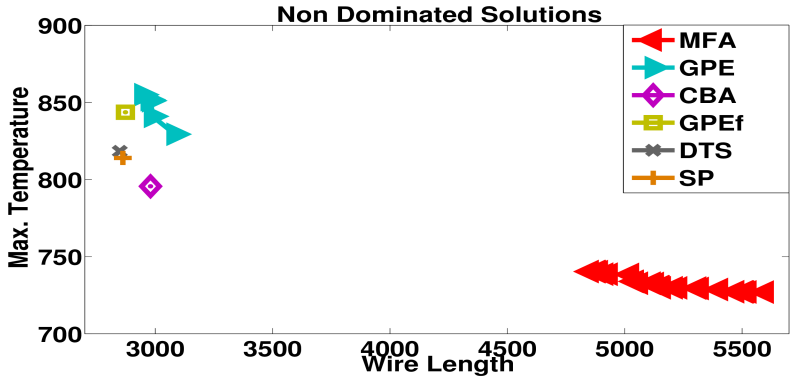
This analysis has showed that MFA produces thermally optimized solutions as compared to the rest of representations. However, the thermal impact has been measured with an approximate model. In order to study the feasibility of the obtained solutions, a thermal analysis with exact simulators is required.



(a) 48 cores scenario



(b) 64 cores scenario



(c) 128 cores scenario

Figure 5.6: Optimized solutions found with MFA, GPE, CBA, DTS, and SP in the 48, 64, and 128 scenarios

Thermal Analysis

We present here the thermal analysis performed in the 48 and 128 cores scenarios, corresponding to the smaller and the larger architectures.

	Wire L.	T_{MAX}	T_{MEAN}	$Grad_{MAX}$
48 orig.	6733	411.82 K	344.29 K	109.75
48 opt.	6875	345.30 K	331.75 K	31.81

Table 5.1: Thermal response of the 48 cores configurations

48-cores configuration We compare an optimized configuration of the 48-cores heterogeneous platform to the original 48-cores Niagara platform represented in Figure 5.7. This configuration corresponds to a regular arrangement of the components of the architecture. As a consequence, the SPARC cores (SPC) are placed above the others producing hotspots. On the other hand, Figure 5.8 shows the thermal maps of the different layers of a nondominated solution returned by the thermal-aware floorplanner. This figure shows an optimized placement of the SPARC cores (SPC), Power6 cores (P6), memories (L2) and crossbars (Cross) achieved by the floorplanner. In this configuration the hottest elements (SPARC cores) are generally placed in the borders of the chip and in the outer layers, separated as much as possible. In fact, the floorplanner avoids placing cores above the others as vertical heat spread is also taken into account. The crossbars are placed in intermediate layers to minimize the wire length.

The metrics considered for the thermal analysis of these two platforms are the maximum and mean temperature of the chip and the maximum thermal gradient. In Table 5.1 we present the thermal response of these two different configurations. These results show that our floorplanner proposes thermally optimized configurations. The peak temperature of 411.82K found in the original configuration is reduced to 345.30K while the mean temperature is reduced in 12.54K. We can see that the maximum thermal gradient of the optimized configuration is reduced from 109.75K to 31.81K. Therefore, not only the temperature of the chip is reduced but it is also more evenly distributed. On the other hand, the wire length of the optimized configuration is a 2.11% greater than the original which translates into a small performance penalty.

128-cores configuration For this larger configuration, we analyze one of the optimal floorplans obtained with our parallel implementation. Figure 5.9 shows the

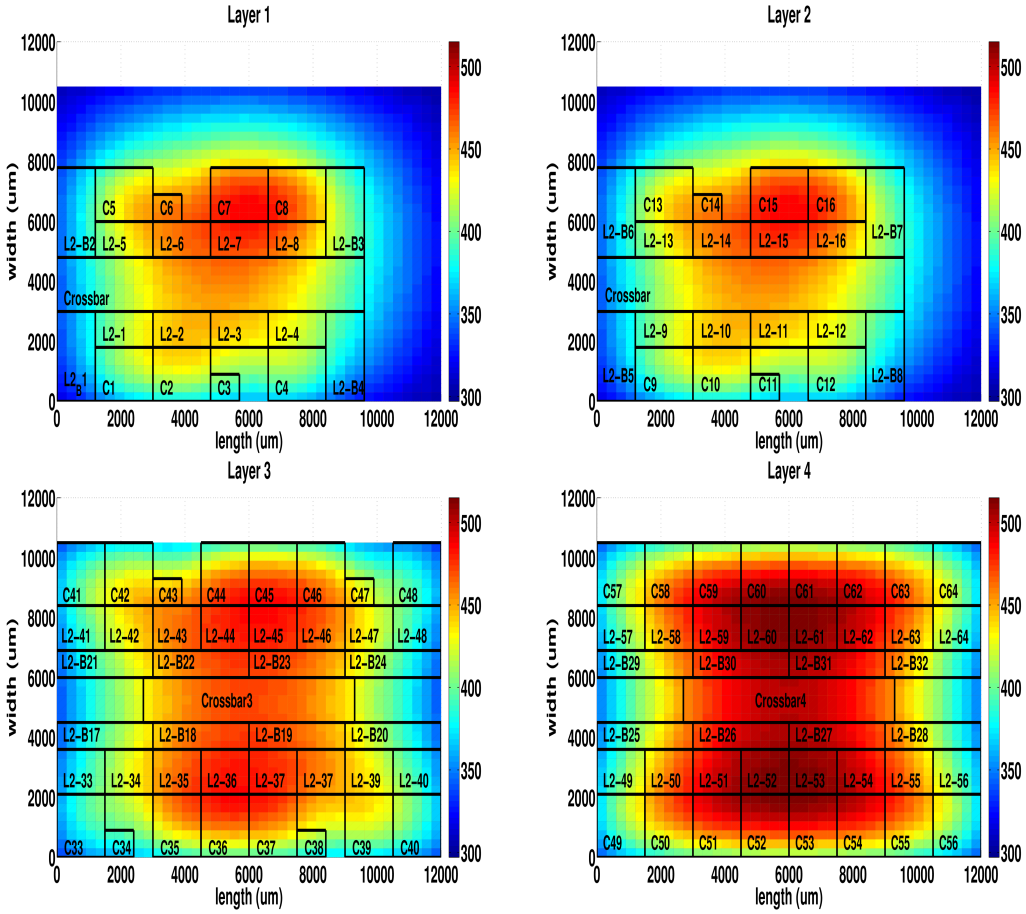


Figure 5.7: Thermal map of the 4 layers of the baseline configuration of the 48 cores platform

thermal map of the chosen solution. As for the 48-cores platform, we can see that the SPARC cores tend to be placed in the outer layers and at the borders of the chip. The memories and the crossbars are placed in the inner layers. This way both chip temperature and wire length are minimized. Nevertheless hotspots appear in this configuration. Table 5.2 shows the thermal response of an optimized configuration of the 128 cores platform. Note that the results obtained from the simulation of the original 128 cores configuration are not presented here since the achieved temperatures are extremely high, making such configuration not appropriate as benchmark for comparison. The hotspot visible in the first layer of the chip corresponds to the peak temperature of the chip reaching 396.84K. The mean temperature is 362.50K while the maximum thermal gradient is 75.80K. Given that the obtained working temperatures are still far from the safety region, further research and simulations

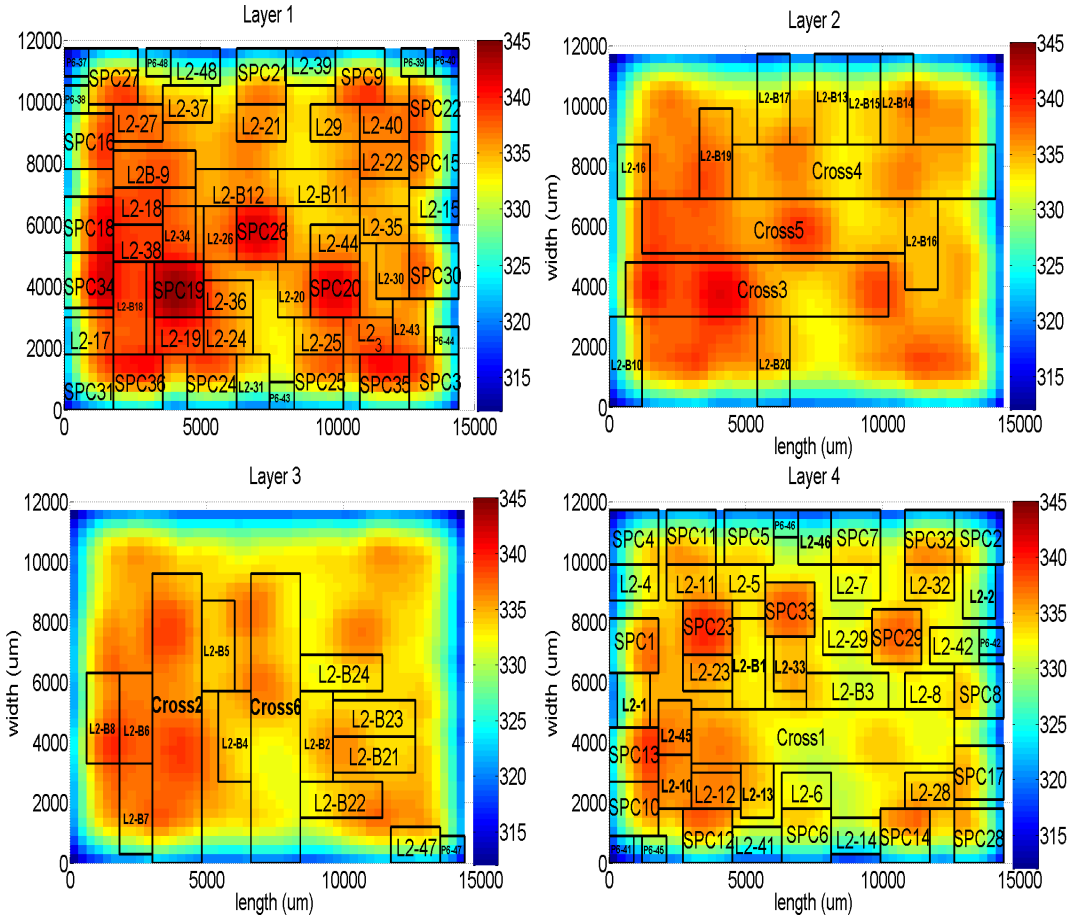


Figure 5.8: Thermal map of the 4 layers of a nondominated solution of the 48 cores platform

with cooling techniques are required to study the feasibility of these architectures.

	Wire L.	T_{MAX}	T_{MEAN}	$Grad_{MAX}$
128 opt.	31587	396.84 K	362.50 K	75.80

Table 5.2: Thermal response of the 128 cores configuration

5.1.3 Conclusions

Current and short term future 3D many-core architectures require thermal-aware floorplanning techniques able to reduce peak and mean temperatures. In this section, it has been verified that MFA provides optimized configurations for systems

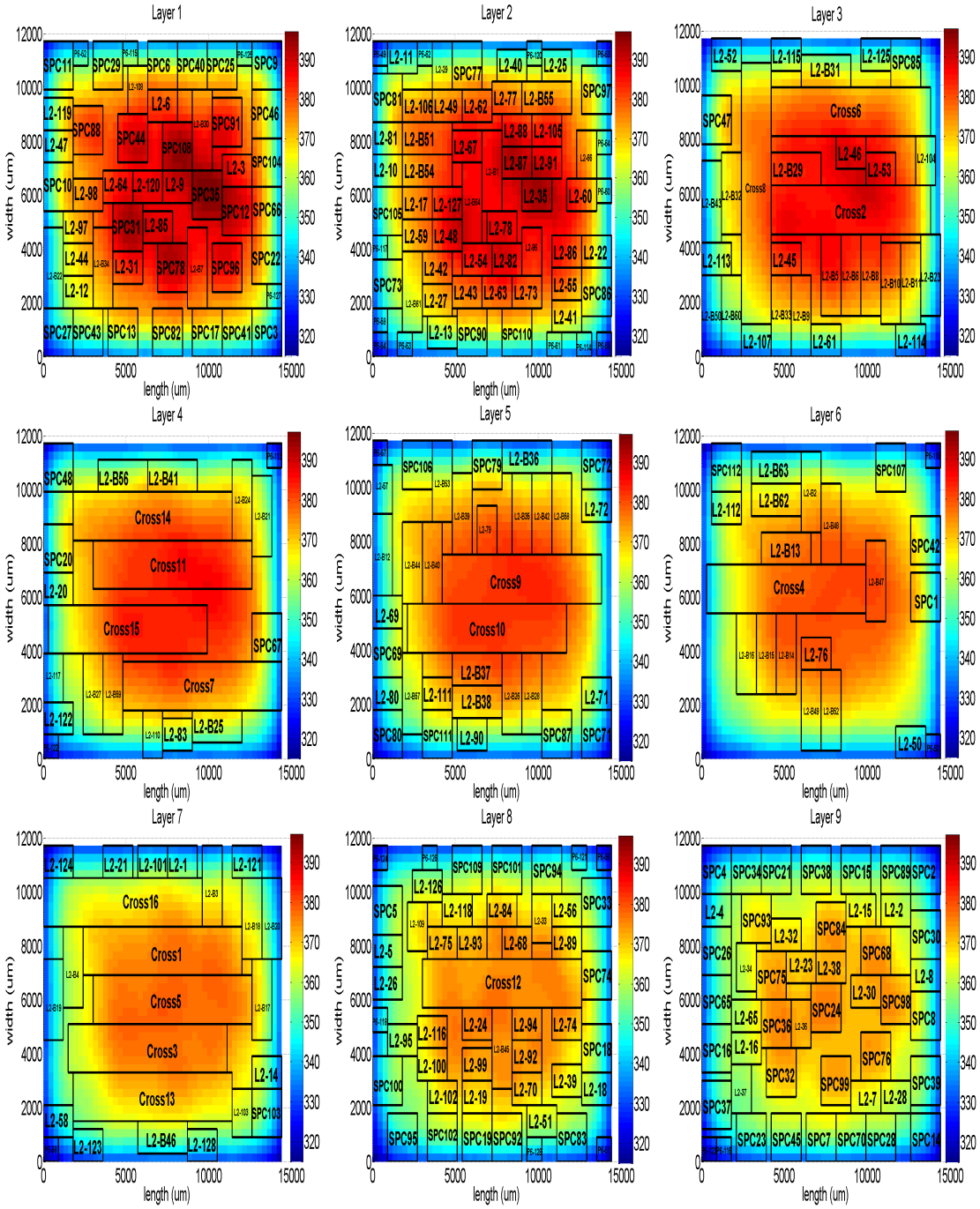


Figure 5.9: Thermal map of the 9 layers of a nondominated solution of the 128 cores platform

composed of 48 and 128 heterogeneous processor cores. Moreover, the performed study of the convergence of the floorplanner shows an appropriate behavior of the algorithm in the optimization process. However, current floorplanning techniques such as MFA that take into account thermal issues spend the most of the execution time dealing with decoding and evaluation of solutions. It is therefore necessary to speedup the floorplanning process to allow for a more efficient exploration of the solution space.

5.2 Master-Worker Model Parallel Floorplanner

We present a master-worker model implementation for MFA, described in the previous section. The effort of parallelization is justified as the evaluation phase of the algorithm takes over 99% of the execution time. This bottleneck appears because every individual of the population has to be decoded and thermally evaluated in every generation of the process.

5.2.1 Details of Parallelization

The master-worker model is used because, even though the fitness is based on a simplified thermal model, the computational cost of this evaluation increases quadratically with the number of components. Therefore, it is interesting to exploit the fact that evolutionary algorithms are intrinsically parallel and carry out the evaluation of the population in a concurrent manner. Figure 5.10 depicts the approach used in this work.

The master distributes the population among n workers, splitting the computational load in n ways so it does not carry out any evaluation. Once the workers have finished their task, they send the outcome together with the received population subset to the master. Although the algorithm stops and waits for all workers to finish, it is clearly much faster than the sequential execution as long as each subset is large enough for compensating communication times.

We propose a multi-threaded implementation where only the master executes the main thread of the algorithm. Since only the workers execute the evaluation of

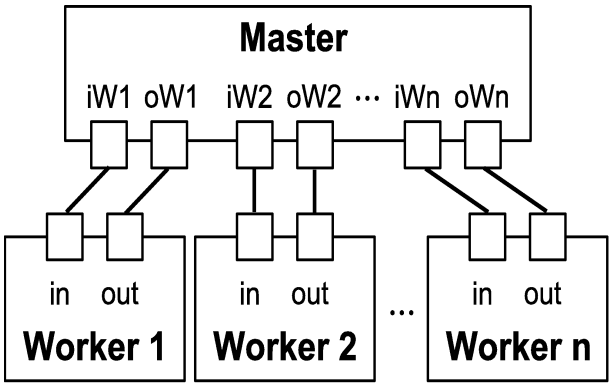


Figure 5.10: Master-Worker configuration

different subsets of the population, a speed-up similar to the number of cores in the processor that executes the algorithm is expected.

5.2.2 Experimental Setup

The experimental work will analyze the speedup obtained with the parallel version of the MOEA while making clear that the quality of the solutions remains the same.

We consider the 48 and 128 cores architectures presented in Section 3.2.2. For the sake of clarity, their main features are rewritten here and summarized in Table 5.3. The first architecture is composed of 48 processor cores: 36 SPARC and 12 Power6 cores; 72 memories and 4 crossbars for inter-processor are added summing up a total of 124 components. In the second architecture, 128 cores are included: 96 SPARC plus 32 Power6. In addition, 191 memories and 9 crossbars are considered, therefore 328 components need to be placed in this scenario. The floorplanner will place the processors, the local memories and the crossbars in 4 and 9 layers respectively.

Platform	Sparc	Power6	Memories	Crossbars	Total	num. Layers
NH48	36	12	72	4	124	4
NH128	96	32	191	9	328	9

Table 5.3: Description of the Niagara-based architectures

5.2.3 Results

In the first analysis, we study the speedup obtained with the parallel version of the floorplanner. Then, the results obtained with the parallel and sequential version are compared to validate the master-worker approach.

Speedup Analysis

We aim to find the optimal number of workers leading to the maximum speedup. Note that each worker is mapped to a different thread. To this end, we perform a parametric sweep of the number of workers, from 1 to 9, both in the 48 and 128 core scenarios. In order to obtain the execution time of these optimizations, we run each one of the worker configurations five times, obtaining the average execution time and speedup for both scenarios. The experiments were carried out in a dedicated Intel Core-i5 machine, a 4-core processor, running at 2.80GHz. We set a fixed population size of 100 individuals and 250 generations as the MOEA parameters for the 48 core scenario optimization. Table 5.4 shows the average execution times and corresponding speedups for these runs, with a number of workers ranging from 1 to 9.

# workers	1	2	3	4	5	6	7	8	9
time (s)	24171	12398	8904	6918	6380	6421	6665	6502	6386
speedup	1	1.95	2.72	3.49	3.79	3.76	3.63	3.72	3.79

Table 5.4: Average execution times and speedups obtained in the 48 cores scenario

Figure 5.11 shows the obtained speedups in the 48 core scenario. As can be seen, the speedup increases almost linearly until the number of workers reaches the number of cores in the processor used for these optimizations (4-core processor).

In our master-worker scheme, the particular set of individuals assigned to a given worker determines its execution time. Then, if the worker receives a set of individuals that need more time to be evaluated, the worker will slow down the overall process. On the contrary, if the individuals need less time to be evaluated, the worker will be idle until all workers finish their assigned task. Therefore, the optimization follows this behavior: the population is divided into as many sets as declared workers, then

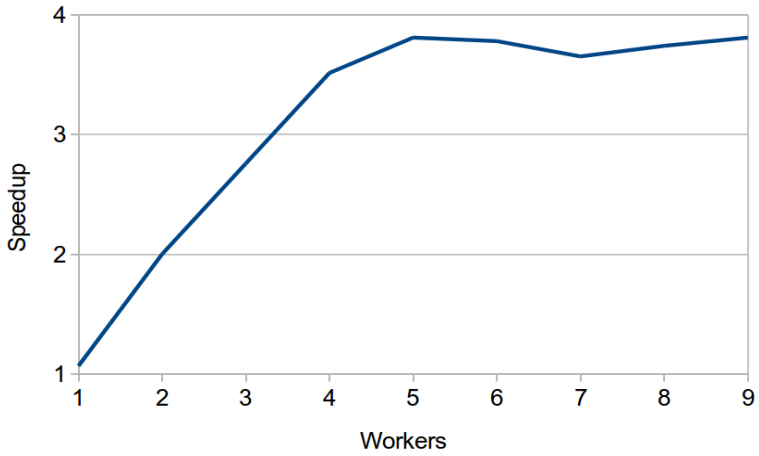


Figure 5.11: Average speedup values obtained in the 48 cores scenario

each set of individuals is sent to a different worker and the evaluation begins. Once the workers finish their evaluation task, they send the computed results to the master and remain idle until the slowest worker has finished. As a result, the slowest worker establishes the execution time of the evaluation of each generation.

Each worker is run in a different thread, and the load assigned to one thread cannot be divided into different processor cores. As a consequence, if the number of workers is higher than 4, the operating system scheduler will distribute the execution of the threads among the 4 cores. Then, the cores will swap between different threads, advancing on the execution of each one. As can be seen in Figure 5.11, results for 5 workers and above present an asymptotic trend on speedup because the usage of resources is already maximized. The particular case of the 5 workers configuration obtains the maximum speedup because it maximizes the resources and, at the same time, avoids unnecessary swaps between threads. These results confirm that the parallelization of the evaluation phase with the master-worker scheme contributes to significant speedups. In addition, there is no remarkable penalty due to the parallelization, because the speedup values above 4 cores tend to be similar.

In order to strengthen this hypothesis, the same tests were run for the 128 cores scenario, where the evaluation time for each individual is much longer. Here, the number of generations of the MOEA has to be scaled up because the number of components is increased. Therefore we consider a number of generations equal to the

total number of components, which is 328: 128 cores, 191 memories and 9 crossbars. The population size remains set to 100 individuals. Table 5.5 shows the speedup obtained in this scenario, considering a number of workers ranging from 1 to 9. Figure 5.12 displays the speedup trend for these data.

# workers	1	2	3	4	5	6	7	8	9
speedup	1	1.47	2.14	3.11	3.19	2.55	2.69	2.54	2.86

Table 5.5: Speedups obtained in the 128 cores scenario

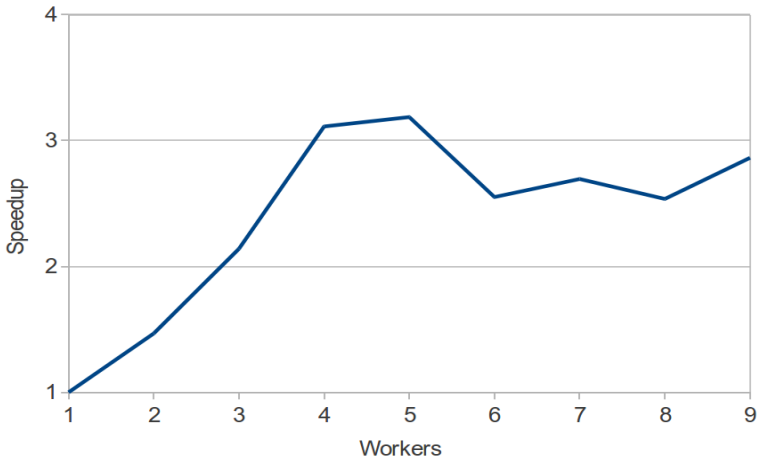


Figure 5.12: Average speedup values obtained in the 128 cores scenario

The 128 cores scenario presents the same behavior as the 48 cores one. The resources of the CPU are maximized from the 5 workers configuration on, and higher numbers of workers obtain similar speedup values. However, the performance improvement is lower than in the 48 cores configuration. This behavior occurs because the individual evaluation time is much higher in this 128 cores scenario, and the execution time of the threads does not differ so much. In the 48 cores case, the available processor slots, due to the different execution time between threads, allow the evaluation of more individuals than in the 128 cores configuration. As a consequence, the workers waiting for free processor cores advance more in their execution, obtaining higher speedups. On the contrary, the workers of the 128 cores scenario

are not able to exploit the processor free slots to evaluate as many individuals as in the 48 cores case, therefore obtaining lower speedup values.

Validation of the Solutions

Since EAs are intrinsically heuristic, two executions will not produce exactly the same results unless the seed of the used random generators is previously fixed. Hence, in order to prove that our proposal is valid it is necessary to define a measure that analyzes the outputs (solution sets) both from sequential and parallel executions.

Such a measure is usually referred to as *Indicator* I. In this work the *Hypervolume* indicator, proposed by Zitzler and Thiele [162], has been used. The hypervolume $I(A)$ gives a measure of how much of the objective space is 'covered' by an obtained solution set A; returning the hypervolume of that portion of the objective space that is weakly dominated by A. To this end, the objective space must be bounded. Otherwise a reference point that must be at least weakly dominated by all solutions in A is used. Finally, higher values of I correspond to higher quality of the measured set.

The comparison has been carried out between the sequential execution, the 4-workers and the 5-workers versions of the parallel implementation. This choice was motivated because these configurations had obtained the highest non-saturated speed-ups. The results shown in Figure 5.13 were obtained after running 30 optimizations, each one with 250 and 328 generations in the 48 cores and 128 cores scenarios respectively. As expected, the three boxplots inside each figure show a similar outcome; with 25th and 75th percentiles almost identical within the 48 and 128 core plots.

5.2.4 Conclusions

In this section we have presented a parallel implementation using a master-worker scheme of MFA, a thermal-aware Multi-objective Evolutionary Algorithm for 3D floorplanning. This model provides optimized configurations for systems composed

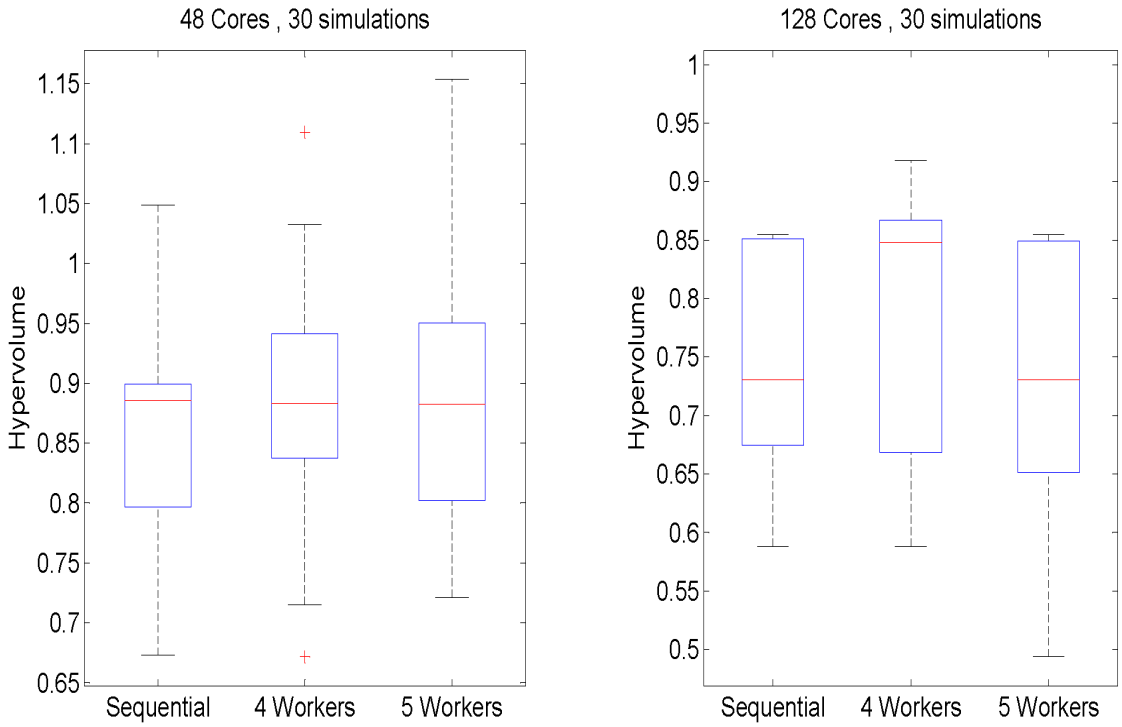


Figure 5.13: Hypervolumes for 48 cores (left) and 128 cores (right); both measured in the sequential and in the parallel execution, the latter with 4 and 5 workers. The central line is the median, the edges of the box are the 25th and 75th percentiles, the whiskers extend to the most extreme data points not considered outliers, which are plotted individually (+ mark)

of 48 and 128 heterogeneous processor cores in a reduced time. We have shown that the highest speedup values are obtained when the number of workers is closer to the number of cores of the processor that runs the algorithm. In our experiments, run on a 4-core processor, we have obtained maximum speedup values of 3.79 and 3.19 respectively for the 48 and 128 cores platforms with 5 workers.

As shown in Section 5.1.2, the thermal optimization achieved with MFA leads to a significant temperature reduction. However, in the larger 128 cores scenario, the obtained configurations are not feasible since a prohibitive peak temperature of $396.84K$ is reached. Therefore, chip temperature must still be reduced. To this end, we propose to incorporate additional knowledge to the optimization by means of a power profiling step previous to the floorplanning process.

5.3 Power Profiling-Guided Floorplanner

In this section, we add knowledge to the thermal-aware floorplanning problem by means of a power profiling step. In fact, the temperature of a given chip depends on physical factors such as the power dissipation of the processors, the size of the memories etc. but it also depends on the dynamic profile of the applications. One of our contributions is to consider energy profiles based on the simulation of real world applications executed on powerful MPSoCs. In fact, this problem is generally approached considering only the worst case scenario in terms of power dissipation. The power profiles are obtained with OVPsim [76], which is a high level multiprocessor simulator for architectural exploration.

5.3.1 Power Profiling Methodology

Studied Architectures

In this case, we do not considered the Niagara architectures presented in Section 3.2.2 as OVPsim does not fully support the simulation of Power6 processors. In this case, we study different 3D many-core architectures composed of memories and SPARC, ARM CORTEX-A9 and POWERPC 440 9SF processors. We study three scenarios that differ from each other in the number and distribution of cores. In the first architecture there are 30 cores with a small proportion of low-power processors: 20 SPARC, 5 CORTEX-A9 and 5 PPC440. The second platform is composed of a medium number of cores with an homogeneous distribution: 22 SPARC, 22 CORTEX-A9 and 22 PPC440 adding up to a total of 66 cores. Finally, in the last scenario, there are 129 cores: 43 SPARC, 43 ARM and 43 PPC. This case corresponds to a scaled up version of the 66 processors architecture. In all cases, there is a shared memory common to all the processors (used for the inter-processor communication) and a local memory for every core. The size of the local memories must be small, as manycore architectures with big local memories are not easy to fabricate. The components composing the three proposed architectures are summarized in Table 5.6.

Platform	Sparc	Cortex-A9	PPC440	Memories	Total	num. Layers
P30	20	5	5	31	61	3
P66	22	22	22	67	133	4
P129	43	43	43	130	259	5

Table 5.6: Description of the three studied architectures

Application Benchmarks

We work with ParMiBench [77] which is composed of parallel versions of typical applications. We select 11 applications grouped into six different categories: Calculus, Network, Security, Office, Multimedia and Mixed. Therefore we have 6 different benchmarks corresponding to several applications that will exhibit very different execution profiles. As a result, the power profiles obtained are different from one benchmark to another. These benchmarks are simulated on bare machines. When an operating system is considered, the power consumption, and hence the temperature tends to be homogeneous due to the execution of the kernel. Therefore, we decide to avoid this overhead in our power profiles. The benchmarks need to be adapted for execution with OVPsim (mapping the compiled code into specific memory regions, replacing the system calls etc.). To obtain the profiling statistics for a given benchmark, we assign a parallel application and a shared memory region to each of the groups of processors working together. Table 5.7 shows a task distribution example where 7 groups of six processors compute a Dijkstra shortest path algorithm and two groups of 12 processors compute a Patricia algorithm. The IDs in this table correspond to processor IDs. The processors in the same row form a group that will execute a given parallel application. With these task distributions, we aim to simulate scenarios that could happen on a real platform. Further details of the task distributions used to obtain the profiles of the different applications are given in [7].

To compute the power dissipated by each of the processors and memories of the studied architectures, we perform simulations of the 3 platforms and 6 benchmarks (18 simulations). We split the simulation of a given benchmark in time slices called windows and study the evolution of the dissipated power versus time for every element. For each of these windows we count the executed instructions and the

Application	SPARC ID	CORTEX-A9 ID	PPC440 ID
pat1	0,1,2,3	22,23,24,25	44,45,46,47
pat2	4,5,6,7	26,27,28,29	48,49,50,51
dijk1	8,9	30,31	52,53
dijk2	10,11	32,33	54,55
dijk3	12,13	34,35	56,57
dijk4	14,15	36,37	58,59
dijk5	16,17	38,39	60,61
dijk6	18,19	40,41	62,63
dijk7	20,21	42,43	64,65

Table 5.7: Task distribution of the Network benchmark for the 66 core architecture

idle cycles per processor, as well as the read and write accesses per memory (local and shared). We set the window period to 128ms. This value is chosen to be long enough to reduce the impact on performance of the profiling phase, but short enough to capture the dynamic behavior with the required accuracy.

Memories

We set the size of the local memories to $512KB$ and the size of the shared memory to $4MB$, $8MB$ and $16MB$ for the 30, 66 and 129 cores platforms respectively. We consider both the local and shared memories to be direct mapped SRAM memories, with a block size of 64 bytes and a transistor size of 45 nm. We obtain the energy consumption and area values with the CACTI software [67]. The energy per write access E_w value is approximated by $E_w = E_r * 1.5$, where E_r is the energy per read access (see [67]).

Processors

To obtain realistic power profiles of heterogeneous platforms, we have chosen three processor architectures with a different computing power. In fact, we are specially interested in understanding the effect of synchronization and communication in the temperature of the chip. Figure 5.14 shows a typical power dissipation pattern of three different processors working together. We can see clearly how the activity of the SPARC core changes periodically over time, waiting for slower processors.

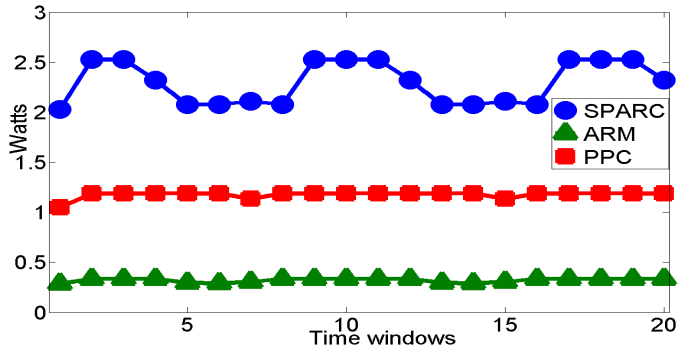


Figure 5.14: Common power consumption pattern caused by synchronization

We assume that the energy consumption of a given processor depends on its working frequency and its state. We consider two states: active or idle. To compute the power densities of the processors, we need their areas and a power consumption value for both the active and idle states. In [121] we find that the power consumption of the SPARC is 4W at 1.4GHz. In the case of the CORTEX-A9, we find that 0.4W is the estimated power dissipation working at 830 MHz while the PPC440 9SF dissipates 1.1W at 667MHz (see [6] and [72]). As in [10], we approximate the power dissipated in the idle state with $P_{idle} = P_{active}/10$. We consider the following areas: $3.24mm^2$, $1.5mm^2$ and $6.2mm^2$ for the SPARC, CORTEX-A9 and PPC440 respectively (see [121], [6] and [72]).

Power Profiling Overhead

Obtaining these power consumption profiles represents a significant overhead in the optimization process. However, such process must only be executed once and is obtained in a preliminary phase of the floorplanning process. As can be seen in Table 5.7, the different tasks are assigned to independent groups of processors working together. By combining or replicating the activity of these reduced groups, it is possible to obtain the profiles of new architectures of varied sizes. For more details of the methodology involved in the retrieval of the power profiles, the reader is referred to [7].

5.3.2 Multi-Objective Evolutionary Algorithm

The validated MFA explained in Section 5.1 is used to perform the temperature and performance optimization. The multi-objective optimization targets the following objectives:

1) The first objective is determined by the topological relations among placed blocks. It represents the number of violated topological constraints (overlapping between placed blocks).

2) The second objective is the wire length approximated as the Manhattan distance between interconnected blocks.

The following objectives $(3, 4, \ldots, m)$ are a measure of the thermal impact, each one based on a profile of power consumption. We use the simplified thermal model presented in Section 2.5.2 in which both the power densities of the different blocks and the contribution of their neighbors are considered. Thus, our remaining objectives can be formulated as:

$$J_{k\in 3..m} = \sum_{i<j\in 1..n} (dp_i^{k-2} * dp_j^{k-2})/(d_{ij}) \tag{5.1}$$

where k refers to a power profile, dp_i^p is the power density of block i for power consumption p, and d_{ij} is the Euclidean distance between blocks i and j.

In our case, we obtain up to 600 different power consumption values (100 time windows $\times$ 6 applications). Obviously, 600 objectives is too high for a MOEA, since it will converge too slow or will not converge at all. Therefore, we discuss how to reduce this number of objectives in the next section.

5.3.3 Experimental Setup

The experimental work will analyze the thermal optimization achieved by the floorplanner in the three different scenarios presented in Section 5.3.1 (30, 66 and 129 cores architectures). The floorplanner will place the processors, the local and shared memories of the 3D manycore platforms in 3, 4 and 5 layers respectively. For each of the three scenarios, we obtain four different floorplans:

1) As we do not have any original configuration to compare with, we propose as baseline a performance optimized floorplan targeting only the feasibility and the wire length. From now on, this configuration will be called BAS.

In order to obtain the thermally optimized floorplans, it is not possible to take directly into account all the data retrieved from our simulations. In fact, if the power dissipation of every element for every benchmark and time window was considered, the floorplanner would target 600 objectives and would hardly converge. Therefore, to obtain the other three floorplans, we consider different power metrics computed with the data retrieved from the simulation of 100 time windows for the 6 different execution profiles.

2) The first one of the remaining configurations is obtained considering the mean power dissipation for each element and profile (AVG). Then, the floorplanner looks for feasible solutions that minimize six thermal objectives (one per profile) and the wire length.

3) Another configuration is obtained considering only the highest power consumption per element (WOR). Hence, three objectives are targeted: feasibility, a thermal objective and the wire length. This case corresponds to the strategy followed by other thermal-aware floorplanners.

4) Finally, a weighted sum of the power consumptions of the different profiles (all weights are equal) is considered for each element (WSM). In this case, the algorithm targets feasibility, a thermal objective and the wire length.

We run the evolutionary algorithm with a population of 100 individuals and 500 generations. The crossover probability p_c is fixed to 0.90 and the mutation probability p_m to $1/\#blocks$ (see [46]). We consider a fixed area equal to the total sum of the areas of the different elements. This value is increased in a 15% corresponding to the minimal area overhead necessary for wire routing. The configurations obtained are chosen among a front of nondominated solutions returned by the floorplanner. In all the studied cases, we select the configuration that minimizes the wire length.

5.3.4 Results

As explained in the previous section, we obtain four different floorplans (BAS, AVG, WOR and WSM) for each of the three considered scenarios (30, 66 and 129 cores platforms). In this section, we compare the thermal profiles of these different configurations. To evaluate them, we propose two experiments. The temperatures presented in this work are given in degrees Kelvin and obtained with the thermal simulator 3D-ICE presented in [138] (see Section 2.5.2).

Thermal Response in the Worst Case

In the first experiment, the obtained floorplans are evaluated with the highest power dissipation per element. This case corresponds to the worst scenario. The metrics considered for the analysis of the experimental results are the mean and maximum temperature of the chip and the maximum thermal gradient. These metrics are usually found in thermal-related analysis. In Table 5.8, we present the thermal profiles of the four different configurations.

$\#cores$	30			66			129		
	T_{max}	T_{mean}	Gr_{Max}	T_{max}	T_{mean}	Gr_{Max}	T_{max}	T_{mean}	Gr_{Max}
BAS	416.30	355.22	108.44	440.24	357.35	136.10	443.15	361.56	137.57
AVG	394.80	350.10	84.93	410.187	352.41	101.92	396.33	355.32	87.06
WOR	388.42	349.03	73.46	401.27	349.55	91.22	414.45	355.98	104.59
WSM	387.68	349.73	74.12	403.34	349.40	91.742	400.44	354.63	91.75

Table 5.8: Worst case

The results show that our power profiling-guided floorplanner produces thermally optimized configurations. The hotspots found in the performance optimized floorplans (reaching 443.15K) justify the thermal optimization presented in this work. Compared to the baseline, we can see that in all the cases our floorplanner reduces the peak and mean temperatures and the thermal gradient. Therefore, not only the temperature of the chip is reduced but it is also more evenly distributed. For example, we can appreciate that the dramatic peak temperature of 416.30K found in the baseline is reduced to 387.68K in the WSM configuration of the 30 core platform. We illustrate this example in Figure 5.15 where we show the thermal maps of

the first layer of the BAS and WSM configurations. In the baseline configuration, the floorplanner tends to place the processors near their local memories ignoring the presence of hotspots. In the other, the hottest elements (the SPARC cores) are separated as much as possible, generally placed at the borders of the chip. As a consequence, we observe a much better thermal response of the WSM configuration. Vertical heat spread is also taken into account. Hence, the floorplanner avoids placing cores above the others. In both cases the shared memory is placed in the second layer to minimize the wire length.

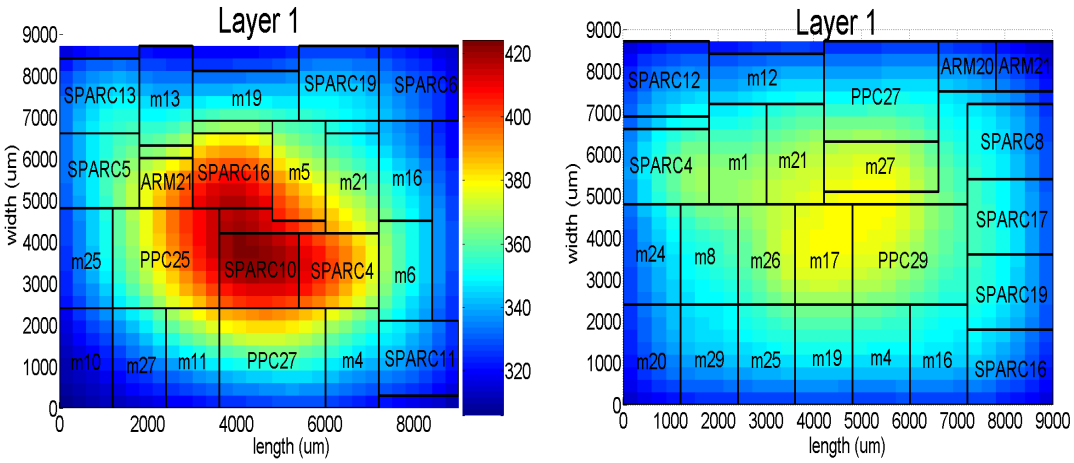


Figure 5.15: Thermal map of the first layer of the 30 cores BAS(left) and WSM(right) configurations

The baseline configuration (BAS) is not an acceptable solution, in fact it reaches peaks of 443.15K in the 129 cores configuration. Moreover, the peak temperature increases with the number of cores in all of the cases. Therefore, from now on we will use the baseline to compare the wire length of the thermally optimized floorplans, but we will not study its thermal response. Choosing a configuration among the thermally optimized floorplans is not immediate. It is due to the fact that some of these configurations present a better performance (wire length) while others have a better thermal behavior. We propose a selection criterion in section 5.3.4.

Thermal Response for the Different Benchmarks

In the second experiment, we evaluate the same floorplans in a more realistic way. The thermal behavior of the different configurations is simulated for 100 time windows with the power dissipation values obtained from our execution profiles. Three

metrics are considered in this case to compare the different configurations. The mean of the maximum temperatures of the different time windows is computed as well as its standard deviation. This metric is a good indicator of the existence of hotspots in the studied chip. We also compute the overall mean temperature of the chip and the mean of the maximum thermal gradients as well as their respective standard deviations. Table 5.9 shows the metrics retrieved for the six studied benchmarks in the 30, 66 and 129 cores scenarios. We now comment the results obtained in the three different scenarios.

30 cores scenario The results obtained for the peak temperatures are not conclusive as none of the configurations clearly outperforms the others. The highest difference is obtained in the case of the first Benchmark where the temperature reached by the WSM is 2.8K and 5.02K lower than the one obtained with the WOR and AVG configurations respectively. The mean temperatures are very similar for all the configurations. In fact there is a maximum difference of 1.12K between the different configurations (Benchmark 5). As for the peak temperatures, a first analysis of the maximum thermal gradients does not lead to an immediate selection of the best configuration.

66 cores scenario The peak temperatures obtained in the WSM are lower than those obtained in the AVG configuration saving up to 7.22K. Only in the case of the Benchmark 1 the peak temperature of the WOR configuration is the lowest one, with a difference of 1.63K over the WSM. On the other hand, selecting WSM leads to a maximum reduction of 2.63K in the peak temperature for the Benchmark 3. Once again, there are no significant differences between the mean temperatures of the studied configurations. In the case of the maximum thermal gradients, the results are similar to those obtained for the peak temperature: the WSM outperforms the other configurations in five out of six cases, reducing up to 10.05K the gradient of the AVG (Benchmark 5) and 3.09K the gradient of the WOR (Benchmark 3). In the case of the Benchmark 1, the best thermal gradient is obtained for the WOR configuration, with a difference of 1.27K and 4.75K with the WSM and AVG configurations respectively.

30						
	Tmax					
	BENCH1	BENCH2	BENCH3	BENCH4	BENCH5	BENCH6
AVG	$340.95_{6.58}$	$342.55_{4.56}$	$340.22_{7.80}$	$310.35_{3.15}$	$348.33_{2.68}$	$343.75_{3.38}$
WOR	$338.73_{7.44}$	$345.51_{2.66}$	$341.12_{11.09}$	$311.43_{2.82}$	$347.48_{2.60}$	$340.29_{3.22}$
WSM	$335.93_{6.30}$	$345.56_{3.79}$	$340.80_{10.02}$	$309.37_{2.89}$	$346.65_{2.56}$	$341.38_{2.63}$
	Tmean					
	BENCH1	BENCH2	BENCH3	BENCH4	BENCH5	BENCH6
AVG	$322.43_{6.14}$	$320.11_{2.26}$	$324.70_{5.59}$	$306.28_{1.95}$	$331.41_{1.65}$	$325.56_{2.86}$
WOR	$322.19_{5.79}$	$319.82_{2.19}$	$324.33_{5.49}$	$306.24_{1.91}$	$330.29_{1.58}$	$324.80_{2.70}$
WSM	$322.50_{6.07}$	$319.96_{2.36}$	$324.64_{5.65}$	$306.21_{1.98}$	$331.37_{1.64}$	$325.49_{2.95}$
	Tgrad					
	BENCH1	BENCH2	BENCH3	BENCH4	BENCH5	BENCH6
AVG	$35.88_{5.83}$	$39.45_{4.58}$	$34.40_{6.32}$	$8.47_{2.63}$	$42.70_{2.43}$	$38.73_{3.60}$
WOR	$31.85_{6.25}$	$40.66_{2.65}$	$32.25_{7.97}$	$9.54_{1.95}$	$36.52_{2.08}$	$31.57_{4.45}$
WSM	$29.36_{4.44}$	$41.99_{3.99}$	$34.05_{8.39}$	$7.03_{2.24}$	$36.59_{2.10}$	$34.18_{2.83}$
66						
	Tmax					
	BENCH1	BENCH2	BENCH3	BENCH4	BENCH5	BENCH6
AVG	$334.17_{9.35}$	$365.43_{7.36}$	$386.49_{3.14}$	$369.64_{1.64}$	$349.52_{3.12}$	$369.16_{4.64}$
WOR	$329.77_{5.60}$	$366.91_{6.16}$	$385.90_{5.75}$	$369.60_{1.94}$	$342.91_{2.63}$	$368.72_{5.68}$
WSM	$331.40_{6.28}$	$364.66_{7.52}$	$383.27_{1.96}$	$369.33_{1.62}$	$342.30_{2.62}$	$367.96_{5.62}$
	Tmean					
	BENCH1	BENCH2	BENCH3	BENCH4	BENCH5	BENCH6
AVG	$319.61_{5.22}$	$325.63_{1.06}$	$338.79_{3.43}$	$324.90_{0.60}$	$328.27_{1.51}$	$324.63_{0.62}$
WOR	$318.36_{4.16}$	$325.91_{0.92}$	$337.96_{3.81}$	$324.77_{0.54}$	$325.23_{1.36}$	$324.51_{0.71}$
WSM	$318.57_{4.40}$	$325.27_{0.96}$	$337.29_{3.18}$	$324.33_{0.55}$	$325.40_{1.36}$	$323.83_{0.65}$
	Tgrad					
	BENCH1	BENCH2	BENCH3	BENCH4	BENCH5	BENCH6
AVG	$29.50_{8.42}$	$62.35_{7.65}$	$80.75_{2.22}$	$66.99_{1.87}$	$44.48_{3.04}$	$66.25_{4.98}$
WOR	$24.75_{3.95}$	$64.35_{6.60}$	$79.99_{5.51}$	$67.35_{2.24}$	$35.35_{2.56}$	$66.42_{6.43}$
WSM	$26.02_{4.93}$	$61.41_{7.94}$	$76.90_{1.23}$	$66.91_{2.02}$	$34.43_{2.38}$	$65.45_{6.59}$
129						
	Tmax					
	BENCH1	BENCH2	BENCH3	BENCH4	BENCH5	BENCH6
AVG	$342.31_{7.94}$	$357.95_{0.78}$	$373.18_{4.97}$	$352.95_{1.50}$	$370.42_{4.13}$	$331.63_{2.46}$
WOR	$354.08_{8.90}$	$363.76_{5.07}$	$382.00_{9.27}$	$352.40_{2.51}$	$387.54_{5.19}$	$338.28_{2.93}$
WSM	$349.02_{4.73}$	$363.99_{0.87}$	$371.20_{4.10}$	$351.14_{1.54}$	$373.87_{4.34}$	$336.10_{2.68}$
	Tmean					
	BENCH1	BENCH2	BENCH3	BENCH4	BENCH5	BENCH6
AVG	$327.32_{5.80}$	$328.68_{1.34}$	$338.78_{4.53}$	$321.70_{1.86}$	$344.24_{2.36}$	$322.20_{1.57}$
WOR	$328.25_{5.95}$	$329.19_{1.49}$	$339.01_{4.77}$	$321.57_{1.89}$	$344.87_{2.40}$	$322.57_{1.61}$
WSM	$327.99_{5.52}$	$328.52_{1.43}$	$338.20_{4.99}$	$320.56_{1.98}$	$344.12_{2.35}$	$322.55_{1.59}$
	Tgrad					
	BENCH1	BENCH2	BENCH3	BENCH4	BENCH5	BENCH6
AVG	$36.08_{6.31}$	$54.77_{1.00}$	$66.81_{3.52}$	$51.02_{1.07}$	$62.04_{3.77}$	$25.86_{2.14}$
WOR	$47.75_{7.79}$	$60.06_{4.79}$	$75.95_{8.28}$	$50.70_{1.97}$	$78.82_{4.80}$	$32.12_{2.56}$
WSM	$43.71_{3.80}$	$61.04_{0.91}$	$65.84_{3.22}$	$49.45_{1.14}$	$66.28_{3.98}$	$31.90_{2.49}$

Table 5.9: Thermal metrics retrieved in the 30, 66, and 129 cores scenarios

129 cores scenario The best peak temperature is obtained for the AVG configuration in four out of six benchmarks saving up to 17.12K and 6.71K comparing with the WOR (Bench 5) and WSM (Bench 1) configurations respectively. The WSM outperforms the others in the remaining two cases, reducing up to 10.80K the maximum temperature of the WOR configuration and 2.98K the one of the AVG. The mean temperatures obtained are again homogeneous. The results for the thermal gradients are similar to the ones obtained for the peak temperature. Globally, the WOR configuration shows a worse thermal behavior than the WSM or AVG. Nevertheless, it is the configuration that presents the best performance (wire overhead), therefore it can not be discarded.

Further analysis is required to obtain the best overall solutions as the achieved temperatures seem to show a random behavior. In the next section, we perform a deeper analysis of the quality of the solutions in terms of wire length and thermal behavior in each of the three scenarios. The proposed method shows that there is an optimal metric to guide the floorplanning of the studied architectures.

Performance/Temperature Tradeoff

As showed before, there is a tradeoff between performance (translated into the extra wiring) and temperature. We propose a method to evaluate the different solutions and select the one with the best overall behavior. To this end, we consider 21 different experiments: 3 architectures $\times$ (worst case + 6 benchmarks). For each of the scenarios and thermal metrics studied (T_{Max}, T_{mean} and $Grad_{Max}$) we establish a confidence interval. In a second step we see whether or not the retrieved metrics fall into these ranges of acceptable values. We observe that the thermal metrics retrieved fit to a normal distribution. The intervals are obtained by adding and subtracting the standard deviation to the mean of the considered metric, resulting in 68% confidence intervals. We apply this procedure in all the studied scenarios to obtain the confidence intervals. However, we only show the intervals obtained in the case of Benchmark 4 in the 30 cores platform as a motivational example:

- T_{Max}: The confidence interval is obtained by adding and subtracting the standard deviation σ_{Tmax} to the mean of the peak temperature of the different

configurations called μ_{Tmax}. The resultant interval is:
$[\mu_{Tmax}-\sigma_{Tmax}; \mu_{Tmax}+\sigma_{Tmax}] \simeq [310.38-1.03; 310.38+1.03] \simeq [309.35; 311.41]$

- T_{mean}: The same method is used to obtain the confidence interval for the mean temperatures:
$[\mu_{Tmean}-\sigma_{Tmean}; \mu_{Tmean}+\sigma_{Tmean}] \simeq [306.24-0.035; 306.24+0.035] \simeq [306.205; 306.275]$

- $Grad_{Max}$: In a similar way, we obtain the range of acceptable values for the maximum thermal gradient:
$[\mu_{MaxGr} - \sigma_{MaxGr}; \mu_{MaxGr} + \sigma_{MaxGr}] \simeq [8.35 - 1.26; 8.35 + 1.26] \simeq [7.09; 9.61]$

These intervals allow to group similar values into the same thermal level. In fact a difference of 1K might not be relevant to decide which configuration is better. On the other hand, stepping from a 15% of wire overhead to 25% would result in a dramatic decrease of the chip performance. Once we have these intervals, we see which metrics fall into these intervals (marked as $\surd$ in the following example) and which ones do not (X). There is a third possibility where the metric considered is even below the confidence interval (marked as $\surd\surd$), which is the most desirable situation as the goal is to minimize the wire length and the different thermal metrics. We perform the analysis of the worst case scenario and the 6 benchmarks for the three considered architectures (30, 66 and 129 cores). Based on the obtained confidence intervals, we decide which is the best configuration for each of the 21 considered cases. Table 5.10 shows the result of this analysis for the Benchmark 4 in the 30 cores scenario. In this example, only the WSM configuration satisfies all the constraints.

30	T_{max}	T_{mean}	$Grad_{Max}$
AVG	$\surd$	X	$\surd$
WOR	X	$\surd$	$\surd$
WSM	$\surd$	$\surd$	$\surd\surd$

Table 5.10: BENCHMARK 4

In order to select the configuration with the best overall behavior, we take into account the thermal response and the wire length of the different configurations (see

Table 5.11). We now summarize the results of this analysis.

#cores	30	66	129
AVG	36.88%	16.19%	24.81%
WOR	23.24%	26.68%	17.69%
WSM	13.15%	17.25%	22.31%

Table 5.11: Wire overhead of the different configurations

30 cores scenario In this scenario, the AVG configuration only presents acceptable values in one of the seven scenarios. The WSM and WOR configurations satisfy all the constraints in six out of seven benchmarks. They offer a satisfactory performance while respecting the different thermal metrics studied in this work. However the WSM offers a better performance as it presents 10.09% less of wire overhead. Therefore the WSM is the best configuration for the 30 cores scenario.

66 cores scenario The WOR configuration offers a poor performance. The AVG presents the best performance but it does not satisfy the thermal constraints in four out of seven benchmarks, including the worst case scenario. Therefore it offers a poor response in extreme situations, leading to hotspots. Only the WSM configuration satisfies all the constraints in all the cases. It presents the best overall thermal behavior while only the AVG presents a slightly better performance (1.06%). Hence, the WSM is also chosen in this scenario as the best configuration.

129 cores The selection of the best configuration is not as immediate as in the previous scenarios. Even though the WOR configuration presents the best performance, it is discarded as the metrics retrieved do not fall in the acceptance intervals for four out of seven benchmarks, reaching 414.45K in the worst case scenario. Both AVG and WSM satisfy all the constraints in all the cases. While the AVG offers a better overall thermal response, the WSM configuration presents a shorter wire length increasing the performance in 2.5%. Therefore, in this scenario, there is a tie between these two configurations.

Globally, the WSM configuration always presents an acceptable performance. In addition, the thermal constraints are satisfied in 20 out of 21 cases. The AVG and WOR configurations only present an acceptable thermal response in 10 and 14 cases respectively. Moreover, these two configurations do not offer acceptable performances: the wire overhead of the AVG reaches 36.88% in the 30 cores scenario while in the 66 cores scenario the WOR configuration presents a 26.68% of extra wire. Furthermore, the convergence analysis shows that using the WSM metric leads to an average convergence time reduction of 43.1% and 36.2% as compared to the WOR and AVG metrics respectively. Therefore, the WSM presents the best overall behavior, outperforming AVG and the traditionally used WOR metric. It is due to the fact that targeting a weighted sum of the power consumption of the different benchmarks has allowed to relax the thermal constraints. This relaxation, in turn, has led to a better exploration of the solution space. As a result, solutions presenting a better overall temperature and performance are found.

5.3.5 Conclusion

This work has proposed an efficient approach that incorporates power-profiling information to guide a thermal-aware floorplanner for 3D multiprocessor architectures. The implementation of the tool with evolutionary algorithms has provided thermal-optimized floorplans as compared with a baseline heterogeneous system. The proposed analysis of the tradeoff between thermal behavior and performance shows that considering the worst power consumption does not lead to optimal floorplans. In fact, the configurations targeting a weighted sum of the power consumption of the different benchmarks present a better overall behavior. Such configuration offers a better thermal response in extreme conditions reducing in 14.01K and 12.84K the peak temperature and the thermal gradient of the chip respectively.

However, this study has also revealed the hardness of including dynamic power consumption information in the floorplanning process. In this work, thermal metrics are retrieved from the dynamic power profiles to guide the process, thus simplifying the problem. In fact, if all the candidate solutions were evaluated with the power traces extracted from all the benchmark applications, the runtime overhead of the optimization would be extreme. Moreover, a reduced number of targeted objectives

is required when working with multi-objective evolutionary algorithms.

This work has showed that a correct guiding of the search process has a critical impact on the quality of the retrieved solutions. Therefore, it is interesting to explore other parameters such as the integrated thermal model as it might lead to a better exploration process.

5.4 The Impact of the Thermal Model

As explained in Section 2.5.2, the thermal models used in related works that evaluate the thermal response of 3D chips present a tradeoff between runtime and quality. In the optimizations performed earlier in this work, the use of an approximated thermal model is motivated by its low computational cost. However, in this section, the proposed thermal optimization is guided for the first time by accurate simulations based on a Neural Network model. The suitability of different thermal metrics to guide the optimization is also compared. This way, we either reduce the runtime of the optimization algorithm or achieve a better thermal behavior, resulting in an optimal tool for architectural exploration and post-design thermal optimization.

5.4.1 Floorplanner

As in the previous sections, we employ the MFA floorplanner. Therefore, the floorplanning problem is approached as a multi-objective problem targeting feasibility, wire length, and temperature. We study the impact of considering different thermal models, namely:

1) $APPROX$: The approximated thermal model explained in Section 2.5.2. $APPROX$ is the model used in the previous sections.

2) $3D-ICE$: The exact thermal simulator explained in Section 2.5.2.

3) $NNTM$ The Neural Network Thermal Model introduced in Section 2.5.2. Note that both CPU and GPGPU implementations of this model are available.

5.4.2 Experimental Setup

The purpose of the experimental work is threefold. First, we study the feasibility and the performance impact caused by the integration of an accurate thermal simulator in our floorplanner. Then, we compare the temperature and wire length of the solutions obtained with the different thermal models explained in Section 2.5.2. Finally, we compare the thermal behavior of the configurations obtained targeting different metrics. In particular, we minimize the maximum and mean temperature of the chip and analyze the resulting thermal gradient. To this end, we consider the 30 and 66 manycore heterogeneous architectures proposed in the previous section and summarized in Table 5.12. Note that, in the 129 cores scenario, exact approaches are not feasible due to their dramatic execution time. Therefore, the larger 129 cores architecture has not been included in these experiments.

Platform	Sparc	Cortex-A9	PPC440	Memories	Total	num. Layers
P30	20	5	5	31	61	3
P66	22	22	22	67	133	4

Table 5.12: Description of the two studied architectures

A cell size of $600\mu m \times 600\mu m$ is considered. The die size is fixed to $9600 \times 9000\mu m$ and $12000 \times 11400\mu m$ for the 30 and 66 cores architectures respectively. The power inputs considered correspond to the WSM case (weighted sum of the power consumptions of different profiles) as considering these inputs leads to optimal solutions (see Section 5.3). The neural network is trained with a *proximity* parameter set to $6000\mu m$ and $10000\mu m$ respectively (see Section 2.5.2).

5.4.3 Results

In this section we study the impact of integrating each of the two following models in our floorplanner: original 3D-ICE and NNTM. To this end, we compare the runtime and the thermal optimization achieved using these two models with the standard MFA guided by the approximated thermal model (APPROX).

Performance Analysis

The thermal simulation together with the decoding of the solutions is the bottleneck of the MFA thermal-aware floorplanner taking more than 99% of the execution time. Therefore, the performance of our floorplanner is directly related to the time consumed by the thermal evaluation. Figure 5.16 shows the execution time of the different implementations in the 30 and 66 cores scenarios. The experiments are carried out with an Intel Core-i5 composed of 4 cores running at 2.80GHz.

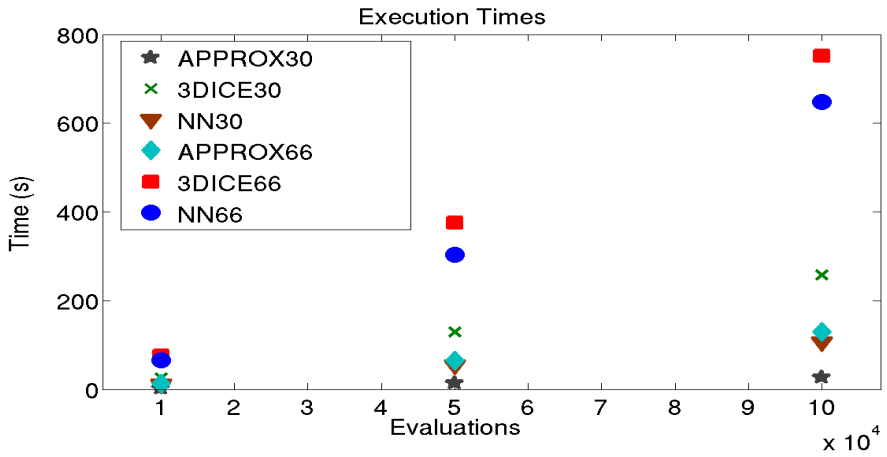


Figure 5.16: Execution Time of the different implementations of the thermal evaluation in the 30 and 66 cores scenario

We can see that in all cases the execution time grows linearly with the number of evaluations. The approximated thermal model is the fastest in the 30 and 66 cores scenarios, followed by the NNTM and the 3D-ICE. In fact, the long execution times required by the original 3D-ICE version justify the usage of the NNTM presented in this work. For example, in the 30 cores scenario the evaluation using the NNTM performs 2.42 times faster than the original 3D-ICE. However, the complexity of the NNTM increases with the size of the studied architecture. Moreover, the complexity of the Neural Network is determined by the number of thermal cells and the chosen proximity parameter (see Section 2.5.2). As a consequence, the speedup obtained with the use of this model in the 66 cores scenario is reduced to 1.15. Nevertheless, the NNTM admits a massively parallel implementation which allows a dramatically faster execution to alleviate the impact of the number of cores.

We propose to use of a GPGPU version of the NNTM to speedup the evaluation phase. Figure 5.17 compares the execution time of the approximated model, the CPU version of the NNTM and a CUDA [119] implementation of the NNTM running on a graphics processor unit. The device used in this test is a NVIDIA GeForce GTX570 composed of 480 CUDA cores and 1280MB of GDDR5 memory. In the 30 cores scenario, the evaluation running on GPU is 8.54 times faster than the CPU version and 2.14 times faster than the approximated model. This speedup is increased to 16.4× and 2.82× respectively in the case of the 66 cores architecture. Therefore, the GPU implementation of the NNTM performs faster than any other of the studied thermal models. Furthermore, the speedup obtained with this massively parallel version increases with the size of the architecture.

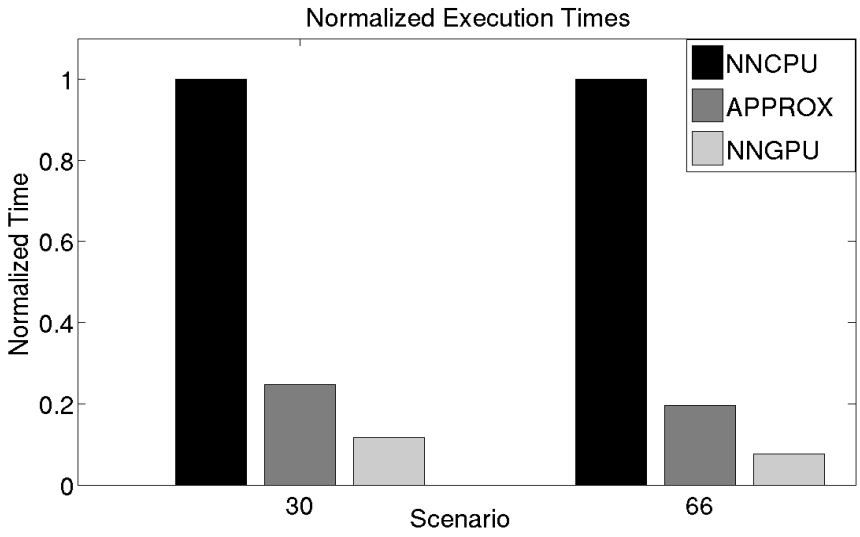


Figure 5.17: Execution Time of the evaluation run on CPU and GPU in the 30 and 66 cores scenario working with the NNTM

The performance analysis has shown that replacing an approximated thermal metric with an exact model such as 3D-ICE has a severe impact on performance leading to a non efficient tool. On the other hand, a significant speedup can be achieved with the introduction of a GPU implementation of the NNTM. The next section studies the thermal behavior and the performance of the solutions obtained with the integration of the 3D-ICE and NNTM model in the floorplanner as compared to the approximated model.

Thermal Analysis

In this section we compare the suitability of different thermal metrics and the thermal optimization achieved with the integration of the different thermal models studied in this work: approximated thermal model, 3D-ICE and NNTM. We run the multi-objective evolutionary algorithm with a population of 100 individuals and 250 generations. The crossover probability p_c is set to 0.90 and the mutation probability p_m to $1/\#blocks$.

For each scenario (30 and 66 cores) and thermal model (APPROX, 3D-ICE and NNTM), we perform 20 runs of the algorithm. The solutions obtained are then evaluated with the 3D-ICE thermal simulator. The metrics considered for the analysis of the solutions are the maximum temperature and gradient of the chip and the wire length.

Thermal Metrics: Thermal-aware floorplanning is a multi-objective optimization problem in which both performance and temperature constraints need to be satisfied. To this end, thermal metrics must be employed to guide the optimization process. However, the efficiency of multi-objective evolutionary algorithms and NSGA-II in particular, tends to decrease when the number of objectives increases. Therefore, the selection of the thermal metric used to guide the floorplanner has a great impact on the behavior of the optimized configurations. In this section we compare the optimization obtained targeting two thermal metrics typically employed in thermal-aware design: the maximum and the mean temperature of the chip. The first metric is employed to reduce the effect of severe hot-spots while the latter tries to reduce, in a homogeneous manner, the temperature of the chip. Note that, in the study presented in Section 5.3, several power inputs are considered to guide the optimization while, in this case, different thermal metrics are targeted.

Figure 5.18 shows the wire length, peak temperature and maximum thermal gradient of the configurations obtained with the NNTM using these two metrics. The analysis of the thermal gradient is especially relevant as this metric is directly related to the reliability of these architectures. The results obtained with the 3D-ICE are not shown here because of space issues, however it is made clear in the next section that the thermal optimization achieved with the NNTM and 3D-ICE mod-

els is equivalent. Note that this study is not extensible to the approximate thermal model as it does not provide an insight of the global behavior of the chip. In fact, the approximated model only targets the distance between the hottest components of the architecture.

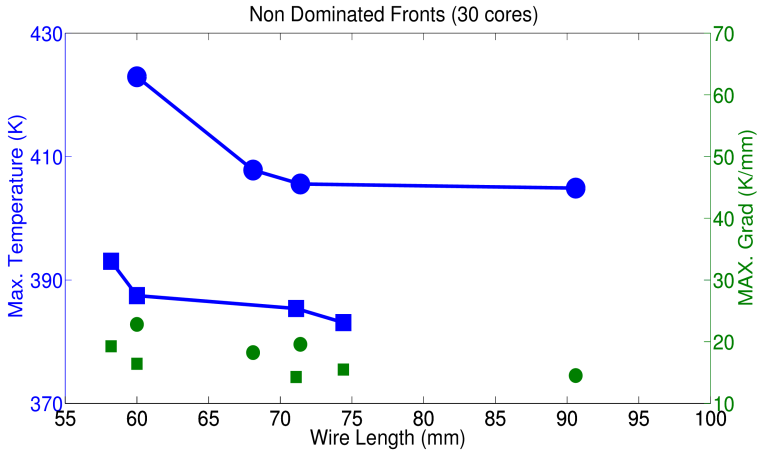


Figure 5.18: Peak temperatures and maximum thermal gradient obtained with the NNTM. The thermal optimization is achieved targeting T_{MAX} (marked with ⊡ symbols) and T_{MEAN} (represented with ◯) in the 30 cores scenario.

To compute the thermal gradient, a series of 100 random power values per component of the architecture is generated. Then, the maximum thermal gradient of a given configuration is retrieved from the thermal simulation of the 100 power inputs. The justification of this procedure is based on the fact that the maximum thermal gradient does not appear when all the elements exhibit a maximum power consumption. In fact, it is when some components are active and others are idle that the largest temperature differences are found.

As can be seen in the figure, targeting T_{MAX} leads to a significant peak temperature reduction. For example, a solution targeting T_{MAX} presents a peak temperature reduction of $35.41K$ as compared to a solution obtained targeting T_{MEAN} with the same wire length ($60.0mm$). In all cases, the thermal gradient remains in acceptable levels. However, we can see that solutions presenting a similar wire length minimize the thermal gradient when the maximum temperature is targeted. For instance, a

solution minimizing T_{MAX} presents a wire length of $71.1mm$ and a maximum gradient of $14.28K$ while a solution targeting T_{MEAN} exhibits $71.40mm$ of total wire and a thermal gradient of $19.58K$. Note that the thermal gradient presented here corresponds to the maximum horizontal gradient of the chip because the higher temperature differences are found between components placed on the same layer. Moreover, in the following we simulate the inclusion of cooling channels that alleviate the vertical thermal gradients.

The performed analysis shows that targeting the maximum temperature leads to a reduction in both peak and gradient temperatures. Therefore, the T_{MAX} metric is used in the remaining of the section.

Thermal Models: We now compare the solutions obtained using the APPROX, NNTM and 3D-ICE models during the optimization process. Figure 5.19 shows the maximum temperature and the wire length of the nondominated fronts of solutions found by the floorplanner using the three different thermal models in the 30 cores scenario. Note that the solutions obtained in the three independent optimization scenarios have been simulated with 3D-ICE at a later stage. In this figure, two different facts can be observed. First, the optimization achieved using the NNTM and 3D-ICE models can be considered similar. However, we have already demonstrated how the NNTM implementation presents lower computational complexity, less simulation time and scales better with the number of integrated cores. Second, the solutions obtained with the 3D-ICE and NNTM models outperform the results obtained with the approximated model. In fact, integrating these accurate models leads to a reduction of the maximum temperature while preserving a similar wire length. In particular, the solution obtained with the approximated model presenting the lowest peak temperature reaches $398.61K$ and exhibits an approximated wire length of $66.6mm$. On the other hand, a solution obtained with the 3D-ICE model presents a maximum temperature of $387.48K$ and a wire length of $62.4mm$ while a solution obtained with the NNTM reaches $387.34K$ and reduces the wire length to $60.0mm$. Thus, the maximum temperature is reduced in $11.30K$ and $11.27K$ while the wire length is reduced in a 6.3% and 9.9% respectively thanks to the more accurate thermal model.

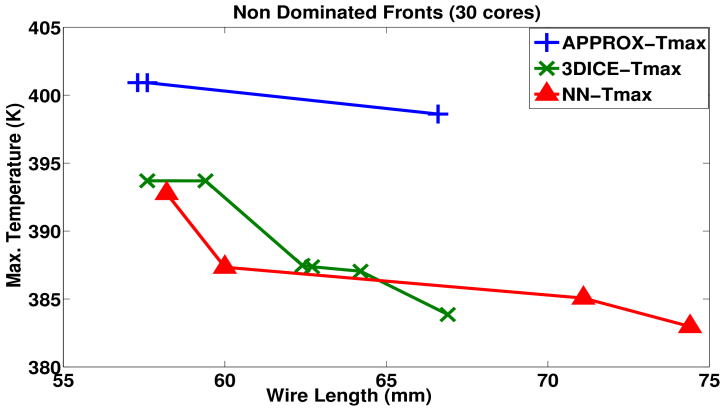


Figure 5.19: Thermal optimization in the 30 cores scenario using the APPROX, 3D-ICE and NNTM models during the optimization process targeting T_{MAX}

Fixed Time Optimization: In the previous section, we have fixed the number of iterations of the optimization process to compare the performance and the thermal behavior of the configurations obtained with the different thermal models. However, as showed in Section 5.4.3, these models exhibit different execution times. In fact, the GPU implementation of the NNTM performs 2.14× and 20.67× faster than the APPROX model and the 3D-ICE run on CPU. In this section, we analyze the thermal optimization achieved with a fixed time deadline. As the NNTM run of GPU is the fastest one, we compare the configurations achieved with the different thermal models at time $T_{NN_{END}}$, i.e. the time needed for the floorplanner to perform 250 generations with the NNTM run on GPU. Figure 5.20 shows the nondominated solutions found with the different models at the referred instant.

In this case, the NNTM leads to a better optimization in peak temperature and wire length than the 3D-ICE. It is due to the fact that in the latter case, fewer iterations of the optimization process have taken place. In a similar way, the differences in wire length between the solutions obtained with the NNTM and APPROX models increase as the minimum wire length obtained with the latter is augmented in a 15.48%.

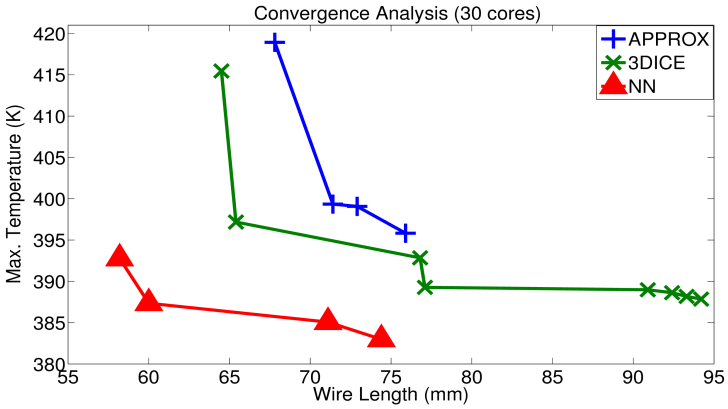


Figure 5.20: Fixed-time optimization of the 30 cores architecture with the APPROX, 3D-ICE and NNTM models

Cooling Cost Analysis: In the previous sections, we have showed that the inclusion of the NNTM in the optimization flow leads to a significant reduction of maximum chip temperatures. However, thermal-aware floorplanning alone is not enough to reduce peak temperatures to acceptable levels. It is, in fact, mandatory to combine floorplanning optimization with other cooling techniques to ensure the feasibility of these architectures. To this end, we consider the inclusion of microfluidic channels in additional layers stacked between the active layers of the 3D chip. Figure 5.21 (retrieved from [138]) shows a traversal view of a four-layered 3D IC with liquid cooling. In the figure, the added layers contain micro-channels placed in a homogeneous manner. In the 30 cores architecture considered in this work, we place 16 channels in each of the two additional cooling layers. The width of the micro-channels is $300\mu m$ and the height is equal to $100\mu m$. A liquid coolant, for instance water, runs through these channels at a fixed flow rate. The higher the flow rate, the higher the cooling effect but also the higher the power needed to pump the water through the channels.

We now analyze through simulations the impact of varying the flow rate on the maximum temperature of two optimized configurations of the 30 cores architecture. The first corresponds to a solution obtained with the approximated thermal model presenting a wire length of $57.3mm$ and a peak temperature of $401.184K$. The

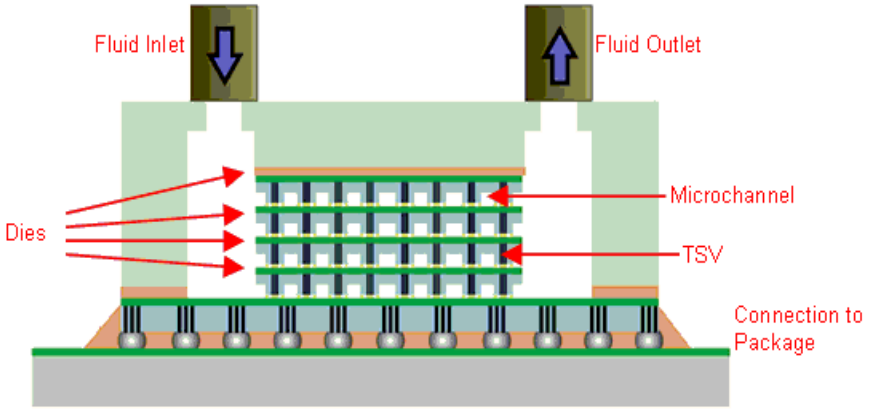


Figure 5.21: Four-layered 3D IC with liquid cooling

second is a solution obtained with the NNTM reaching $387.474K$ and exhibiting a wire length of $60.0mm$. In both cases the selected nondominated solution presents the best tradeoff between temperature and wire length. We also analyze the cooling power consumption needed to reach acceptable levels of temperature. To this end, we retrieve an integrated micro pump power consumption value from [93]. In the cited work, Lee *et. al.* report a micro pump that exhibits a power consumption of $1W$ per microchannel while producing a flow rate of $6ml/min$. In the following, we assume that the power consumption of the referred pump depends linearly on the flow rate. Figure 5.22 shows the cooling effect obtained with a coolant flow rate ranging from 1 to 10 ml/min. Note that the leftmost point of the x-axis (0 ml/min) corresponds to a chip without liquid cooling. The peak temperature of both configurations gradually decreases when the flow rate is incremented. To compare the two solutions, we fix the maximum temperature threshold proposed in [34] to $358K$ (85 °C), and we obtain the necessary coolant rate and the corresponding pump power consumption to meet the constraint. Thus, the liquid cooling power consumption is $0.95W$ in the case of the solution obtained with the NNTM while $1.15W$ are required to cool the APPROX configuration down to an acceptable temperature. Therefore, introducing the NNTM in the optimization flow leads to a cooling energy saving of 17.4%. Note that the low power consumption values presented in this work are increased to unaffordable levels when a higher number of channels is considered. In that case, the presented power saving becomes more significant as it remains proportional to

the required flow rate.

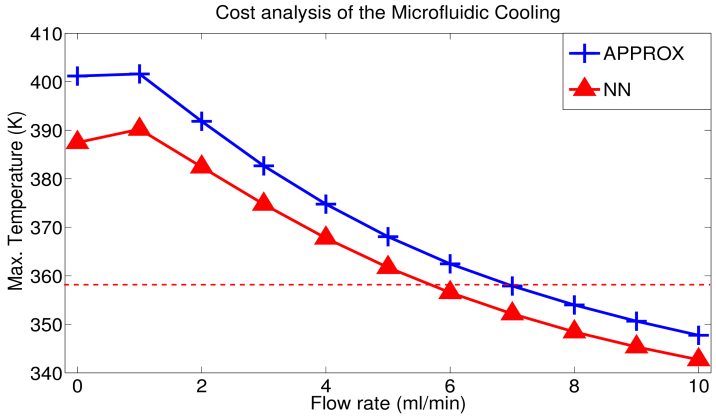


Figure 5.22: Cooling Cost Analysis in the 30 cores scenario

In this scenario, the solutions obtained with the 3D-ICE and NNTM models are more interesting from a designer point of view as both temperature and wire length are minimized. However, the NNTM allows to reduce significantly the runtime of the floorplanning process (see Section 5.4.3) while achieving a similar optimization. Therefore, it is best suited for architectural exploration tasks. Once we have chosen the optimal thermal model for our purpose, we run the optimization of the 66 cores platform.

66 Cores Platform: We show in Figure 5.23 the maximum temperature and the wire length of the nondominated fronts of solutions obtained for the 66 cores architectures. As in the 30 cores scenario, the integration of the NNTM leads to an improvement of the quality of the solutions. In this case, the solution found with the approximated thermal model presenting the shortest wire length ($164.10mm$) reaches $414.27K$. However, this solution presents a performance overhead of 5.85% when compared to a configuration that reaches $412.49K$ found with the NNTM.

The thermal problems encountered for the 30 cores platform are exacerbated here as peak temperatures reach up to $422.08K$. Therefore, it is again necessary to simulate the inclusion of micro-channels to reduce these temperatures down to

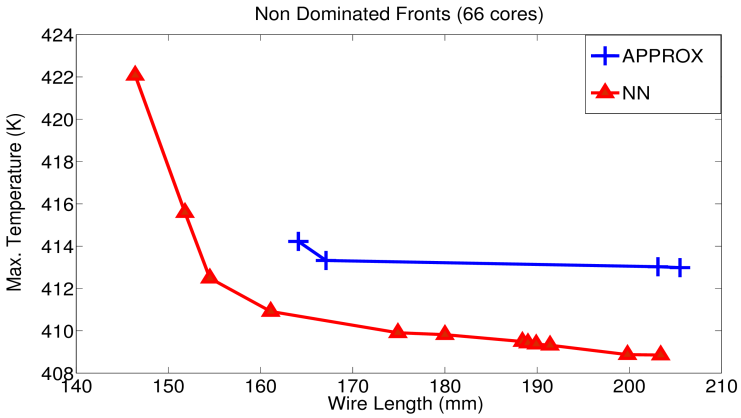


Figure 5.23: Nondominated Fronts returned in the 66 cores scenario

acceptable levels. In accordance with the size of the architecture, we include three additional cooling layers and the number of microchannels is scaled to 19. Next, we analyze the power needed to cool down a solution found with the NNTM reaching $412.49K$ and presenting $154.5mm$ of total wire. Figure 5.24 shows the impact of a varying flow rate in the peak temperature of the referred solution. In this case, the power needed to reduce the maximum temperature to $358K$ increases to $1.75W$.

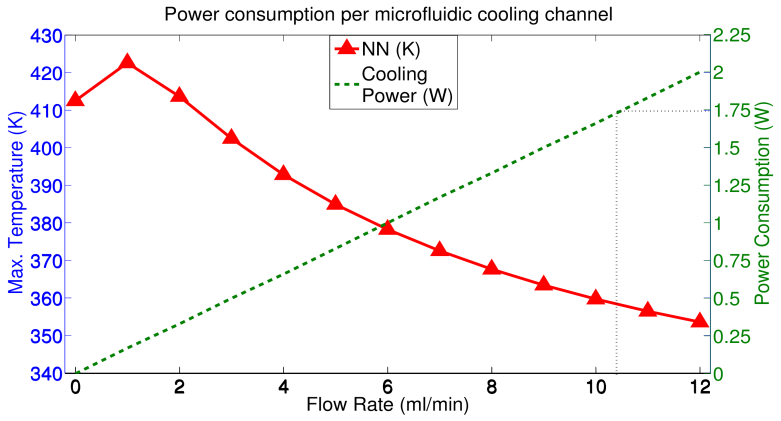


Figure 5.24: Cooling Cost Analysis in the 66 cores scenario

5.4.4 Conclusions

In this section, we have shown that the solutions obtained with the NNTM minimize both temperature and wire length. The thermal optimization achieved is translated into a power consumption reduction of 17.4% as less energy is needed to reduce peak temperature down to acceptable levels. Also, replacing the approximated model with a GPU implementation of the NNTM leads to a speedup of 2.14×. Hence, the NNTM is the most suitable for architecture exploration as it eliminates the tradeoff between accuracy and runtime.

It has also been shown that additional cooling techniques such as the employed microfluidic channels are necessary to achieve feasible temperatures. However, the cooling effect of these microchannels has an impact on the thermal constraints of the targeted architectures that has not been considered in the floorplanning optimization loop. Therefore, considering the new thermal constraints during the floorplanning process will certainly lead to better solutions.

Another major improvement could be achieved with the use of a more suitable representation of the solutions for the 3D thermal-aware floorplanning problem. In fact, most of the floorplanning proposals are based on representations that require time consuming heuristics to the decode the solutions. A new representation allowing a direct mapping of the individuals into configurations of the architecture would alleviate the computational cost of the algorithm as the decoding step would be avoided. Furthermore, it would eliminate heuristics that might limit the exploration space and cause premature convergence problems. Thus, such a representation would be more suitable for fixed-outline floorplanning problems.

Chapter 6

Direct Mapping: a New Representation for 3D MPSoCs

In Chapter 3, it is made clear that the design of modern three dimensional Multiprocessor Systems on Chip (3D MPSoCs) requires new approaches. For instance, when considering 3D MPSoCs, it is common to work with a fixed die size and, more importantly, the thermal impact becomes the most restrictive constraint. In fact, when the thermal constraints are considered, existing representations such as Combined Bucket and 2D Array, Double-Tree and Sequence, Sequence Pair, and Generalized Polish Expression fail to produce thermally optimized configurations as they were engineered to optimize area and wire length of logic circuits. Moreover, the efficiency of these proposals is severely penalized as obtaining the thermal behavior of the different configurations becomes the bottleneck of the different floorplanners. To avoid such overhead, different techniques to speedup the thermal optimization of the targeted architectures is proposed in Chapter 5. First, a Master-Worker implementation of the employed evolutionary floorplanner is presented. Second, a GPU implementation of an exact thermal simulation is used as a guiding cost function in the optimization loop.

However, the speed of the optimization process is not the only factor that should be analyzed to design a floorplanning representation or algorithm. In fact, the search process might benefit from additional knowledge such as the introduced power profiles (see Section 5.3). From a practical point of view, it is also important to provide

designers with an homogeneous toolchain covering all the steps of the fabrication of new 3D chips. In fact, the lack of standards in the design of 3D chips is one of the limiting factors that is delaying the production of these platforms at a larger scale (see Chapter 2). Therefore, new proposals should take into account the valuable contributions and validated tools provided by the VLSI community. For example, the resulting configurations of the floorplanning process should be easily simulated with the state-of-the-art thermal simulators. For instance, RC based thermal simulators such as 3D-ICE [138] split the layers of the chip in grid-like structure, assigning a power consumption value to every "thermal cell". These simulators are rarely used in the optimization step as the resulting process would be highly consuming in terms of time. Nevertheless, there is a widespread use of these simulators as a validation step following the floorplanning process. The problem is that the partitioning performed by these simulators might not correspond to the one used in the optimization process resulting in an inaccurate assignment of power consumption values to the thermal cells. Thus, even if the optimization and validation processes perform correctly independently, the mismatch in the toolchain leads to a lack of accuracy in the validation of the solutions. Therefore, floorplanning tools should provide an output compatible with the state-of-the-art thermal simulators.

In this chapter, we propose a **new representation that leads to outperform the multi-objective evolutionary floorplanner MFA** presented in [38]. In the referred work, the solutions are coded as ordered lists of components and a sophisticated but time consuming heuristic is in charge of the placement of the different elements of the architecture (decoding of the solutions). Such heuristic performs an incremental floorplanning in which the components are sequentially placed in the remaining position minimizing all the targeted objectives. In fact, as the exhaustive heuristic alone is capable of obtaining well performing solutions, the MOEA is designed to obtain the optimal order of the components given to the placing heuristic. Thus, even though a huge search-space of cardinality $n!$ is considered (permutations of the n components of the architecture), the cited approach explores a reduced solution space of the 3D MPSoCs floorplanning problems. Therefore, our insight is that the provided solutions might correspond to suboptimal solutions. Moreover, the heuristic in charge of decoding the solutions might bias the search process. These

two facts lead to reconsider the existing floorplanning proposals in order to provide a useful tool for architectural exploration that satisfies the requirements imposed by current 3D MPSoCs.

6.1 Direct Mapping Thermal-Aware Floorplanner

As in our previous proposals, the thermal-aware 3D floorplanning is approached as a multi-objective optimization problem targeting both temperature and wire length. As MFA, the proposed thermal-aware floorplanner is based on Nondominated Sorting Genetic Algorithm II (NSGA-II), a well-known Multi-Objective Evolutionary Algorithm (MOEA). However, in this case, a different codification of the solutions is employed, namely Direct Mapping (DM).

6.1.1 Representation

As explained in Chapter 2, the thermal-aware floorplanning of 3D ICs can be considered at different abstraction levels corresponding to different design possibilities. The referred 3D approaches range from coarse to fine grained, depending on the granularity of the unit to be stacked, namely:

1. Entire cores and memories
2. Functional unit blocks
3. Logic gates
4. Transistors

Direct Mapping is suitable for both the first (entire cores and memories) and second (Functional Unit Blocks) levels of integration. The proposed representation is suitable for fixed-outline floorplanning problems and allows a direct mapping of the individuals into configurations of the architecture. This is a major advantage as this representation does not require a placing heuristic that might limit the exploration space and cause premature convergence problems in the case of evolutionary algorithms. Furthermore, with the use of this representation, the computational cost of the decoding step is avoided.

Also, this cell-level representation is in accordance with current thermal simulators such as 3D-ICE [138] that split the IC into thermal cells. Therefore, the thermal error due to the different cell sizes used in the optimization and validation processes is eliminated. Thus, a better thermal optimization can be achieved.

In order to work with this representation, the IC is split into a mesh of cells. The choice of the cell size determines the granularity of the problem. In fact, considering big cells leads to a fast optimization process that may speedup the architectural exploration task. On the other hand, the smaller the cell size, the higher the accuracy of the thermal simulations is, but also the higher the complexity of the problem. In particular, floorplanning problems that consider the placement of Trough Silicon Vias (TSVs) are specially challenging as the pitch diameter of the TSVs is generally two orders of magnitude smaller than the components of the architecture. In order to deal with this increased difficulty, some proposals like [37] split the optimization process in two different steps performing the thermal optimization and the TSV placement in a sequential manner. A different workaround is possible adopting a "TSV-as-cell" approach. The proposed representation is compatible with both strategies, and can minimize the area loss in the latter case considering a smaller cell size.

Figure 6.1 shows a configuration and the corresponding representation of an architecture composed of 3 processors (C1, C2 and C3) and 3 memories (L1, L2 and L3). Each component of the architecture is characterized with a coordinate and a boolean flag. The coordinate determines the location of the left-bottom corner of the element while the flag indicates whether or not the given element is rotated. Therefore, the decoding of the solutions is direct and does not require a placement heuristic. In fact, as opposed to other techniques, the exact location of each component is already coded in the chromosome.

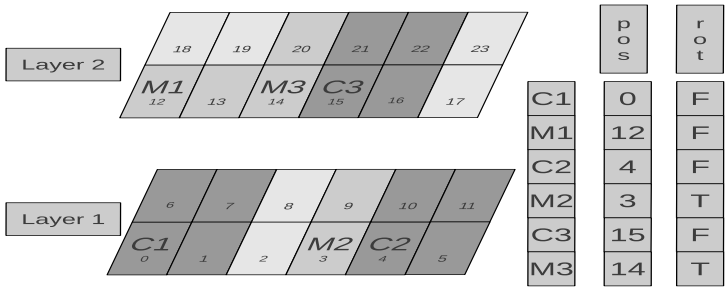


Figure 6.1: Configuration and representation of a simple architecture

This grid-like representation admits a high level of parallelism and is well adapted for execution in massively parallel architectures such as GPUs. Nevertheless, appropriate operators need to be defined to ensure the generation of feasible solutions and the preservation of the diversity of the population.

6.1.2 Initial Population

We need a suitable method to create the initial population managed by the evolutionary algorithm. Given the proposed representation, random coordinates for each component could be generated. However, we have observed that the so-constructed solutions present an elevated number of overlapping components. Thus, they are not appropriate as an initial population. We propose a fast and simple procedure to deal with the initialization of the population in which all the components are sequentially placed in a random order according to a First Fit strategy. The pseudocode of this procedure is shown in Algorithm 2.

```
int pos[numSolutions][numComponents];
bool rotated[numSolutions][numComponents];
int ids[numComponents];
for i ← 0 to numSolutions − 1 do
    ids[i] = i;
end
for i ← 0 to numSolutions − 1 do
    randomShuffle(ids);
    for j ← 0 to numComponents − 1 do
        int id=ids[j];
        int isRotated = rand()%2;
        if (isRotated==0) then
            rotated[s][id] = false;
        else
            rotated[s][id] = true;
        end
        int coord = firstFeasibleCell(s,id);
        pos[s][id] = corners;
    end
end
```

Algorithm 2: Generation of the initial population

The solutions obtained with this method do not necessarily correspond to feasible configurations. However, this placing heuristic avoids overlapping to some extent and the initial solutions present a random behavior. Thus, this method provides a good enough starting point for the optimization process.

6.1.3 Operators

We now describe the employed operators. It is important to note that the crossover and mutation operators have been specifically designed for this representation.

Selection

The selection operator implements a binary tournament strategy. To this end, random couples of individuals are formed and the best solution of each pair is selected. Tuning the size of the tournament selection allows to adjust the selective pressure of the evolutionary process. In our case, the size of the tournament (two) was determined empirically since, with that value, an appropriate behavior of the evolutionary process was observed.

Crossover

We apply an innovative and problem specific crossover operator. In fact, with the use of the proposed representation, typical crossovers such as Single-Point fail to produce feasible offspring because overlapping of different components occurs frequently. As a result, the evolutionary process exhibits premature convergence problems with the use of such operators. The designed operator is based on a Single-Point crossover, in which two children (*Child1* and *Child2*) are obtained from the selected parents *Parent1* and *Parent2*. However, we extend the classical version of the operator to obtain feasible solutions while maintaining most of the genetic information of the parents (see Figure 6.2). To obtain feasible children, we proceed in three steps as follows.

1. First, we randomly select a *point* in $[1..size]$, where *size* is the number of components of the architecture.

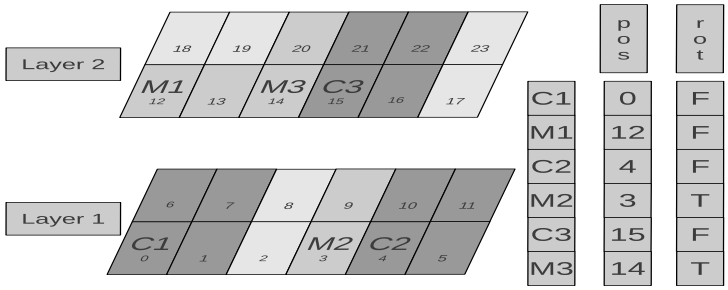
2. Then, the components C_i of *Parent1* and *Parent2* such that $i < point$ are copied into *Child1* and *Child2* respectively.

3. After this first step, the locations of the components C_j such that $j \geq point$ of *Parent1* have to be transmitted to *Child2* while *Child1* must inherit from *Parent2*. If any of these components overlaps with an already placed block, then such component is relocated according to a First Fit strategy.

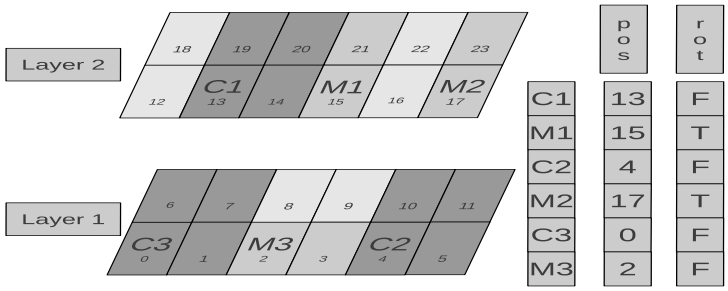
For the sake of clarity, the pseudocode of this procedure is shown in Algorithm 3.

```
int indexPoint = rand() % size;
for i ← 0 to indexPoint − 1 do
    copyComponent(Parent1,Child1,i);
    copyComponent(Parent2,Child2,i);
end
for i ← indexPoint to size − 1 do
    if (feasible(Child2,Parent1,i)) then
        copyComponent(Parent1,Child2,i);
    end
    else if (feasible(Child2,Parent2,i)) then
        copyComponent(Parent2,Child2,i);
    else
        placeFirstFreePosition(Child2,i);
    end
    if (feasible(Child1,Parent2,i)) then
        copyComponent(Parent2,Child1,i);
    else if (feasible(Child1,Parent1,i)) then
        copyComponent(Parent1,Child1,i);
    else
        placeFirstFreePosition(Child1,i);
    end
end
```

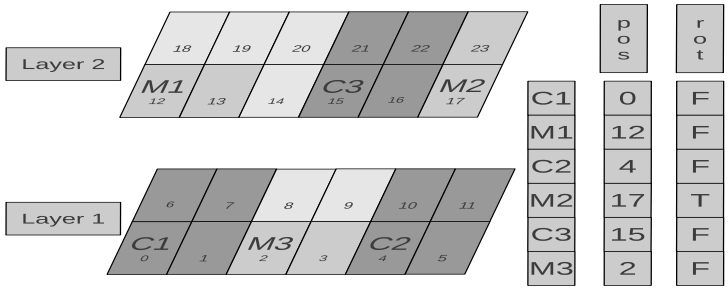
Algorithm 3: Crossover Operator



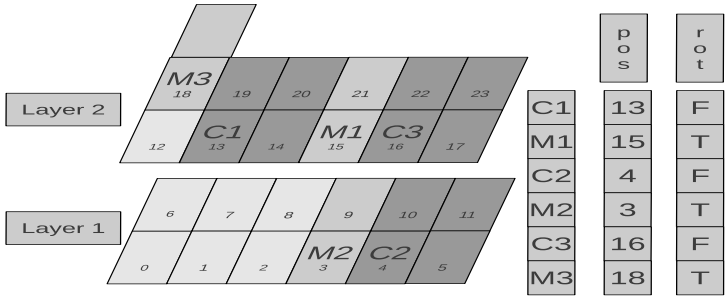
(a) Parent 1



(b) Parent 2



(c) Offspring 1



(d) Offspring 2

Figure 6.2: Crossover operator: A simple example considering a two layer architecture with six components

Mutation

The mutation of the solutions is performed in three ways, each with the same probability:

1. Swapping the position of two elements of the chromosome, resulting in a change of the placement of the two involved components (Figure 6.3(b)).
2. Rotation of a component (Figure 6.3(c)).
3. Randomly moving a component one cell in one of the following directions: up, down, left, right, forward, or backwards (Figure 6.3(d)).

Note that the mutation is not performed if, as a result of the operator, two or more blocks completely overlap or if a block is entirely positioned out of the borders of the chip. However, we allow modules to partially overlap or to be partly placed out of the chip. Such inappropriate configurations are penalized (see Fitness section below) but contribute to increase the diversity of the population and thus to avoid premature convergence issues.

6.1.4 Fitness Functions

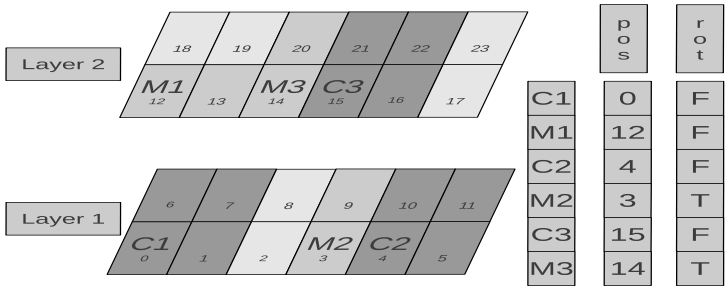
At every generation, the different individuals are evaluated according to the three following objectives:

- The number of violated topological constraints (overlapping between different blocks and area less or equal than maximum area).
- The wire length.
- The temperature of the chip.

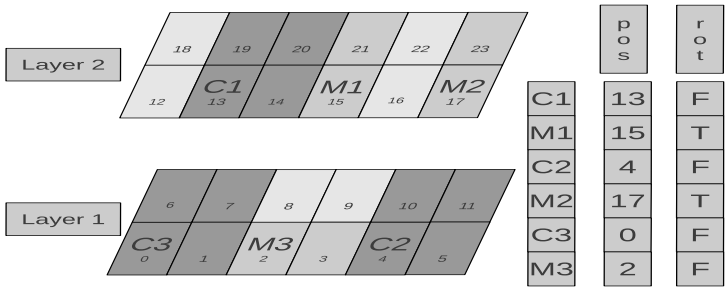
Note that the computation of the wire length and temperature depends on the selected models.

6.2 Direct Mapping vs State-of-the-art Proposals

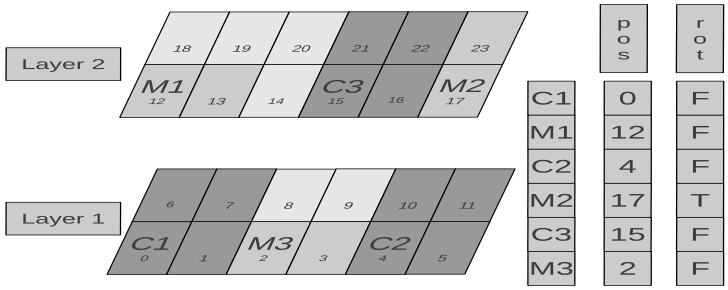
In this section, we compare the described algorithm and representation to existing and validated proposals. To this end, we extend the comparative study presented



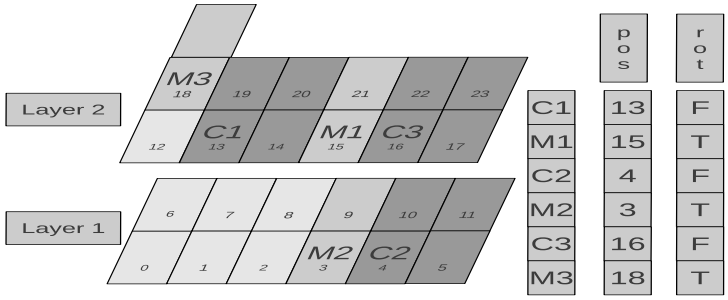
(a) Original Configuration



(b) Swap between components M1 and M3



(c) Rotation of component M1



(d) Component Shift (M1 moved forward)

Figure 6.3: Mutation operator: A simple example considering a two layer architecture with six components

in previous chapters. In Chapter 3, several well-known floorplanning proposals are adapted for the optimization of 3D MPSoCs and their suitability is compared. The results obtained with MFA are added to the comparative study in Chapter 5. In the referred analysis the optimization targeted the following objectives:

Wire Length The wire length is approximated as the sum of the Manhattan distance between interconnected blocks and is computed as follows,

$$W = \sum_{i<j \in 1..n} |x_i - x_j| + |y_i - y_j| + |z_i - z_j| \tag{6.1}$$

where x_i, y_i, and z_i are the coordinates of block i.

Temperature Temperature is measured with the approximated model explained in Section 2.5.2, rewritten here for clarity:

$$\hat{T} = \sum_{i<j \in 1..n} (dp_i * dp_j)/(d_{ij}) \tag{6.2}$$

where dp is the power density of block i and d_{ij} is the euclidean distance between blocks i and j.

6.2.1 Experimental Setup

To validate the optimization achieved with Direct Mapping, we reproduce the experimental conditions described in Section 3.2. Therefore, the same three Niagara-based benchmarks are optimized. For the sake of clarity, the main features of the targeted architectures are rewritten in Table 6.1: number of cores, total number of blocks, length (number of cells), width (number of cells), number of layers, approximated wire length (in cells) and approximated thermal metric. These platforms include an elevated number of SPARC cores exhibiting high power density, thereby exacerbating the thermal issues found in 3D ICs. Smaller Power6 cores are also included to provide a higher degree of freedom to the optimization process. The varied sizes and power densities of the different components help evaluating whether or not the

compared proposals exploit the extra optimization opportunities inherent to heterogeneous architectures.

Benchmark	N cores	# Blocks	Length	Width	Height	W_r	$\hat{T}_r$
NH48	48	124	40	35	4	1320	146.66
NH64	64	165	40	35	5	1636	250.55
NH128	128	328	40	35	9	2889	879.09

Table 6.1: Reference values for the three benchmarks.

It is important to note that, for each of the compared floorplanners, two different strategies are followed to perform the optimization:

1. Starting the optimization from the original configuration of the blocks
2. Starting the optimization from a random configuration of the blocks

Each optimization scenario (benchmark + fixed or random seed) was replicated 10 times. All these experiments were carried out in the same computer used in previous experiments. Thus, the same time deadlines were used to perform the experiments, namely:

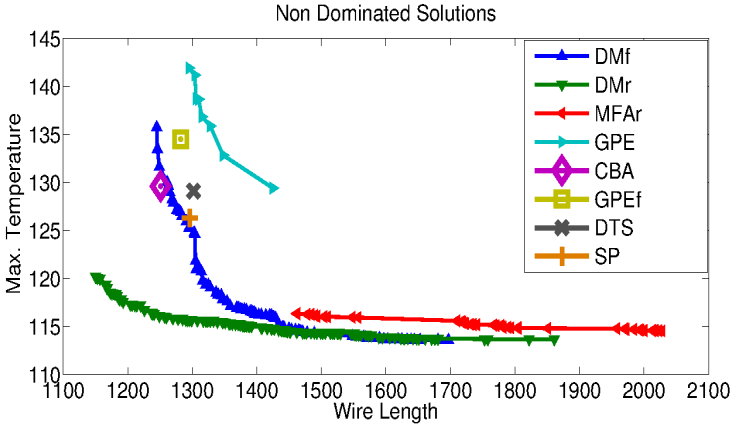
- 12 hours for the 48 cores architecture
- 48 hours for the 64 cores architecture
- 96 hours for the 128 cores architecture

6.2.2 Results

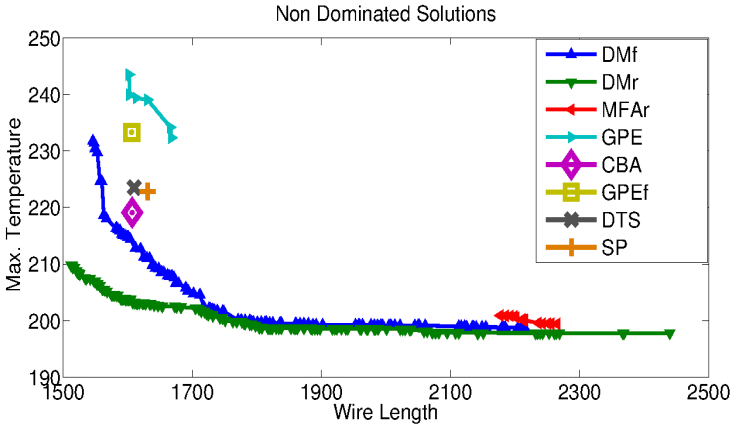
The results of the different optimization scenarios are shown in Figure 6.4. Note that, in each scenario, the front of nondominated solutions is retrieved from all the solutions obtained in the 10 runs of the algorithm.

In the three scenarios, Direct Mapping is the best strategy as each and every solution obtained with any of the other representations is dominated by at least a solution found with DM.

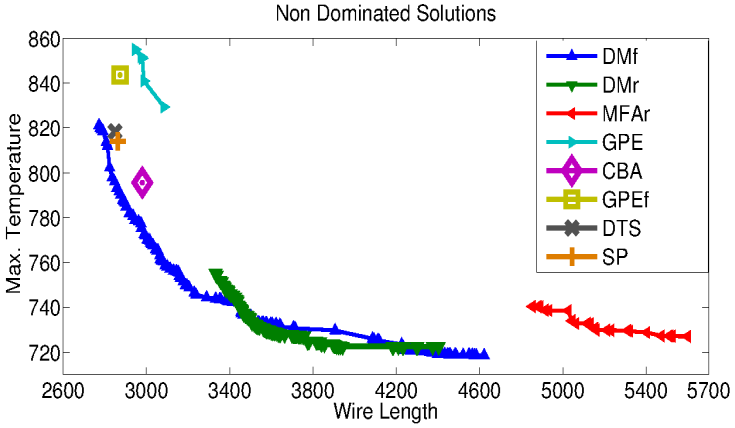
Moreover, the proposed floorplanner returns fronts of nondominated solutions covering a wide range of temperatures and wire lengths. Therefore, it can be said



(a) 48 cores scenario



(b) 64 cores scenario



(c) 128 cores scenario

Figure 6.4: Optimized solutions found with DM, MFA, GPE, CBA, DTS, and SP in the 48, 64, and 128 scenarios

that the implemented multi-objective optimization deals with the tradeoff between the conflicting objectives in an appropriate manner. On the other hand, the other multi-objective proposals, namely GPE and MFA, tend to focus in a limited region of the solution space. Therefore, using DM leads to a wider range of solutions, which represents the ideal case in multi-objective optimization contexts.

Another relevant fact is the difference between the fronts obtained with and without a fixed seed, i.e. whether or not the population was initialized with the original feasible configuration of the benchmark. It is important to note that the configurations obtained with GPE, GPEf, CBA, DTS, and SP all correspond to optimizations starting from the original benchmarks. In fact, no feasible solutions were retrieved when starting with a random seed. Therefore, the results obtained with DM are in accordance with previous experiments as the region of the solution space explored when a fixed seed is set matches the results obtained with other proposals.

We now comment individually each of the three considered scenarios:

48 Cores Scenario

In this scenario, the optimization achieved with a fixed seed (DMf) returns a front that dominates the solutions found with MFA, GPE, GPEf, and DTS. However, no clear domination relationship can be established between DMf and the optimization carried out with CBA and SP. In fact, the solutions obtained with these two representations exhibit similar wire length and temperature as compared to some of the solutions of the DMf front.

On the other hand, starting the optimization from random configurations (DMr) clearly leads to outperform all the other proposals. In fact, the retrieved front is composed of solutions presenting simultaneously shorter wire lengths and lower temperatures.

64 Cores Scenario

The same comments made for the optimization of the 48 cores platform apply to this scenario. In fact, the DMr front clearly presents the best solutions in terms of

temperature and wire length. Furthermore, in this case, the configurations corresponding to the optimization initialized with the original benchmarks (DMf) also dominate the results obtained with all the other proposals. It is probably due to the longer time available for the optimization. In fact, each replica of the algorithm is executed for 48 hours while in the previous scenario the fixed deadline was set to 12 hours. Therefore, the increased size of architecture (from 124 to 165 total blocks) does not match the increment of the running time. It can be said that, in this scenario, the optimization benefits from a proportionally longer optimization phase resulting in a better exploitation of the discovered solutions. Thus, better temperature and performance are obtained.

128 Cores Scenario

The solutions corresponding to the DMf front still dominate the solutions obtained with GPE, GPEf, DTS, and SP. However, in this case, the performed optimization is clearly differentiated from the 48 and 64 scenarios as DMr does not behave as in the previous cases. This front still outperforms MFA, however no clear domination relationship can be established with DMf, GPE, GPEf, CBA, DTS, and SP. In fact, even though a better thermal optimization is achieved, the referred front does not return any solution exhibiting a wire length in the range of the mentioned proposals. This issue might be caused by the procedure in charge of creating the solutions, which might be biasing the search process. In fact, it has already been made clear that the initialization of the population has a great impact on the exploration process. Thus, such initialization method might not be suitable for larger architectures.

6.2.3 Parallelization

Evolutionary Algorithms are intrinsically parallel. In fact, these heuristics typically manage a population composed of several candidate solutions that must be independently evaluated and modified by means of genetic or local search operators. In order to speedup the optimization with a parallel implementation, it is necessary to detect the bottlenecks of the considered algorithm. To this end, we show in Table 6.2 the runtime profile of the sequential version of the algorithm in the 48, 64, and

128 scenarios. Note that the profilings have been replicated 10 times. Thus, the presented values correspond to the averaged times of these executions.

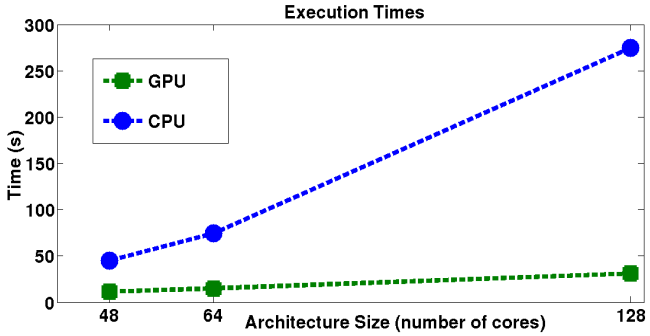
Scenario	48 cores	64 cores	128 cores
Runtime	44.71s	74.63s	274.86s
Init	02.43%	02.72%	02.45%
Selection	00.03%	00.00%	00.00%
Crossover	14.37%	11.64%	06.44%
Mutation	00.17%	00.11%	00.07%
Evaluation	80.07%	83.75%	90.52%
Ranking	02.23%	01.41%	00.39%
Reduction	00.37%	00.37%	00.12%

Table 6.2: Runtime profiling of the proposed floorplanner in three different scenarios. The presented results correspond to the execution of 1000 generations the evolutionary algorithm with a population of 100 individuals

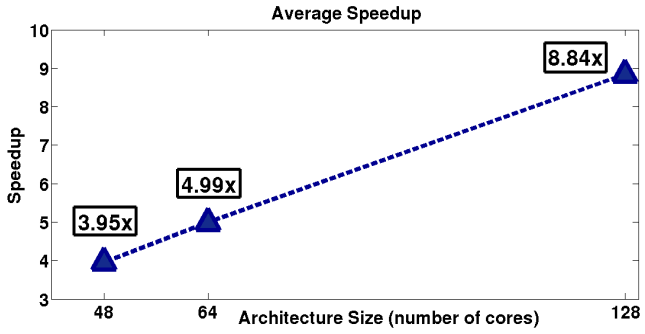
The values shown in the table correspond to the total runtime (in seconds) and to the percentage of the execution consumed by the different methods involved in the multi-objective evolutionary algorithm. It can be appreciated that the evaluation phase of the algorithm consumes most of the execution time in the three cases as it represents respectively a 80.07%, 83.75%, and 90.52% of the total execution time. Moreover, the percentage of the runtime increases with the size of the architecture due to the quadratical computational complexity of the employed approximated thermal model. It is therefore necessary to implement a parallel version of the evaluation phase of this algorithm.

We propose a CPU-GPU version of the algorithm, implemented with CUDA [119]. Note that, only the evaluation step is parallelized as it represents the bottleneck of the algorithm. Moreover, the performed attempts to parallelize other involved methods such as the crossover and the ranking of the solutions resulted in worse performances. For further details of implementation of NSGA-II in CUDA, the reader is referred to [156]. The execution times of the CPU and CPU-GPU versions of the floorplanner in the three scenarios are shown in Figure 6.5(a). Figure 6.5(b) shows the speedup obtained with the parallel implementation. The experiments are carried out with an Intel Core-i5 composed of 4 cores running at 2.80GHz

and a NVIDIA GeForce GTX570 composed of 480 CUDA cores and 1280MB of GDDR5 memory.



(a) Execution Times



(b) Speedups

Figure 6.5: Execution times of the CPU and CPU-GPU implementations of the floorplanner (a) and speedup (b) in the 48, 64, and 128 cores scenarios running 1000 generations of the evolutionary process with a population of 100 individuals

In Figure 6.5(a), the need for the parallel implementation of the algorithm can be appreciated as the execution time of the CPU version increases dramatically with the size of the architecture. On the other hand, the execution time of the CPU-GPU implementation seems to increase linearly when larger architectures are optimized. In fact, as depicted in Figure 6.5(b), speedups of 3.95×, 4.99×, and 8.84× are obtained in the 48, 64, and 128 cores scenarios respectively. The linear increase of the optimization time is crucial as typically, the larger the architecture, the higher the required number of generations is. In larger scenarios, the high number of blocks could saturate the memory available in the GPU device. In such case, the execution time will not increase linearly anymore as more evaluations would be sequentialized. However, such issue does not represent a major drawback

as GPU devices incorporate more and more memory and larger architectures are not expected in the short term future.

6.2.4 Conclusions

A novel representation well adapted for fixed-outline thermal-aware floorplanning of 3D MPSoCs has been introduced. Such representation allows for a direct mapping of the candidate solutions into configurations of the architecture, avoiding the overhead of decoding heuristics that might limit the exploration of the solution space. A multi-objective evolutionary algorithm based on NSGA-II together with representation-specific crossover and mutation operators have been designed to manage the proposed representation.

The adoption of the Direct Mapping representation has led to outperform the results obtained with MFA, GPE, GPEf, CBA, DTS, and SP in three scenarios corresponding to large 3D MPSoCs architectures composed of 48, 64, and 128 cores respectively. The designed algorithm and the corresponding operators have been successfully designed as the performed search returns fronts of configurations covering a wide range of temperatures and wire lengths.

The proposed representation allows to start the optimization process with predesigned initial configurations or with random solutions. The latter has been shown to be beneficial in the 48 and 64 cores scenarios. On the other hand, appropriate initial solutions seem to be necessary to optimize the larger 128 cores architecture.

The comparison performed in this section is fair as all the optimization strategies have been executed with the same time deadline. However, it is beneficial to reduce these elevated optimization times to provide a tool capable of performing architectural exploration tasks in a faster way. To this end, a CPU-GPU implementation of the algorithm has been introduced. This version leads to a speedup of up to 8.84×. More importantly, with the proposed parallel implementation, the optimization time increases linearly with the size of the architecture, resulting in a fast and scalable tool to perform architectural exploration tasks.

6.3 Optimization with Microfluidic Cooling

In the previous chapter (Section 5.4), the suitability of several thermal models in the context of 3D MPSoCs optimization is compared. It is concluded that accurate thermal models such as 3D-ICE or the Neural Network Thermal Model lead to a similar thermal optimization. Additionally, both outperform the results obtained with APPROX, a simplified thermal model.

In the referred section, it is also demonstrated that large 3D MPSoCs architectures composed of highly power consuming components such as SPARC cores reach temperatures above the acceptable range. In fact, cooling techniques are mandatory when dealing with such architectures. Microfluidic cooling channels were included in the thermal simulations of the resulting configurations. This way, peak temperatures were reduced to feasible values.

However, these microchannels were not taken into account in the optimization process. Therefore, the floorplanner could not play with the extra optimization opportunities available in this new scenario. For instance, placing hot components above the others might not lead to severe hotspots if a cooling channel is situated between them in an intermediate layer.

In this section, we compare the thermal optimization achieved with and without the inclusion of microfluidic channels in the thermal simulations performed during the optimization. To this end, we integrate 3D-ICE in the Direct Mapping floorplanner presented in Section 6.1. Note that 3D-ICE allows to simulate 3D stacks with or without microchannels.

To illustrate the targeted 3D chips, we show in Figure 6.6 a four-level 3D chip with intermediate cooling layers. In order to simplify the experimental setup we consider microchannels placed in a homogeneous manner (as depicted in the figure). Moreover, we consider a constant flow rate running through all the channels.

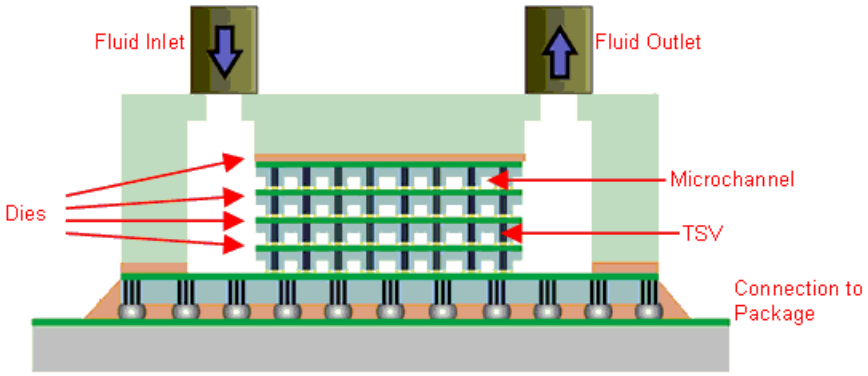


Figure 6.6: Four-layered 3D IC with liquid cooling [138]

6.3.1 Experimental Setup

We perform the optimization of the two smaller Niagara-based architectures used previously with (3D-ICEc) and without (3D-ICE) the inclusion of the cooling channels in the optimization process. Note that these exact approaches are not feasible in the 128 cores scenario due to the dramatic execution time. Therefore, the larger 128 cores architecture has not been included in these experiments. The main characteristics of the studied platforms are summarized in Table 6.3.

Benchmark	N cores	# Blocks	Length	Width	Height	W_r	$\hat{T}_r$
NH48	48	124	40	35	4	1320	146.66
NH64	64	165	40	35	5	1636	250.55

Table 6.3: Reference values for the three benchmarks.

As in previous experiments, the multi-objective optimization targets both temperature and wire length. The latter is approximated with the Manhattan distance between interconnected blocks while the thermal behavior of the configurations is obtained with 3D-ICE. The optimization starts from random configurations as better results were obtained with this strategy (see Section 6.2.2). It is worth noting that the 3D-ICE simulation time is increased when microchannels are considered. Therefore, two series of experiments are proposed.

- First, a fixed-time optimization is performed setting the following time deadlines: 4 and 6 hours for the 48 cores and 64 architectures respectively. This

experiment will show if there is a tradeoff between optimization time and quality of the retrieved solutions.

- In the second series of experiments, the number of generations fixed to 500. This way, we will show whether or not including the microchannels in the optimization loop allows to benefit from the extra optimization opportunities available in 3D chips with liquid cooling.

Three and four intermediate cooling layers are introduced in the 48 and 64 cores platforms. The width and the distance between channels are both equal to $300\mu m$ while the height is $100\mu m$. Each of the cooling layers includes 39 microchannels and the flow rate remains constant at $12.0ml/min$.

Thus, the number of experiments amounts to eight (two platforms + with or without cooling + fixed time or fixed generations). Each optimization scenario is replicated 10 times. All these experiments are carried out with an Intel Core-i5 composed of 4 cores running at 2.80GHz.

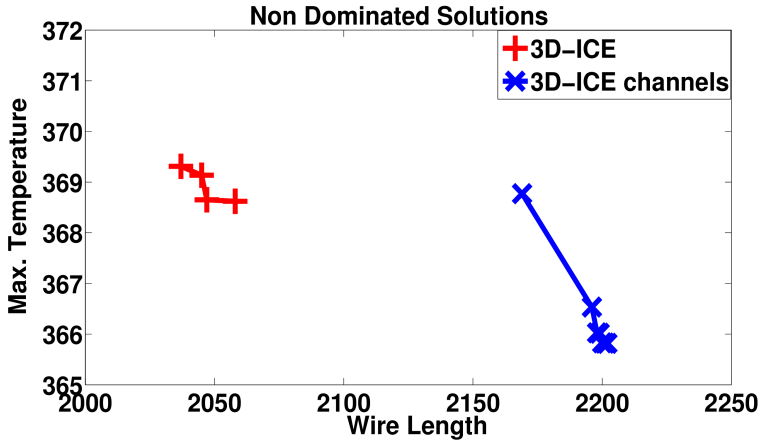
6.3.2 Results

We compare the configurations obtained with the two proposed experimental setups.

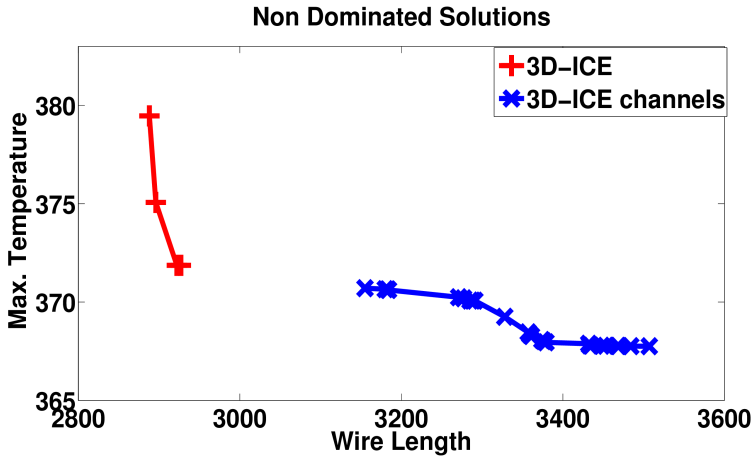
Fixed-time Optimization

Figure 6.7 shows the nondominated solutions retrieved from the 10 replicas of the experiments with the proposed time deadlines. Note that the temperatures shown in this section have been obtained with 3D-ICE including the corresponding cooling layers.

In both scenarios, the thermal optimization achieved considering microchannels leads to a better thermal behavior of the configurations. However, the solutions obtained with the standard 3D-ICE present a better performance. For instance, in the 48 cores scenario, a configuration obtained with the 3D-ICE model without liquid cooling reaching $368.6K$ presents a wire length 5.1% shorter than a solution presenting a peak temperature of $368.8K$ obtained with the inclusion of microchannels in the optimization process. A similar behavior can be observed in the 64 cores



(a) 48 cores scenario



(b) 64 cores scenario

Figure 6.7: Optimized solutions found with and without including microchannels in the optimization process in the 48, and 64 scenarios considering a fixed time

scenario. As in the previous case, the solutions resulting from the incorporation of liquid cooling in the floorplanning process present a lower temperature but also a a longer wire length.

Thus, to achieve an optimal thermal optimization, it is necessary to consider the new thermal constraints resulting from the considered cooling technique. In fact, the severe thermal constraints imposed by large 3D architectures are relaxed as severe hotspots are avoided. Floorplanners must benefit from the higher degree of freedom allowed by the inclusion of cooling techniques. For instance, hot components might be placed on top of each other if a microchannel is placed between them.

The difference in performance might be due the fact that considering microfluidic cooling has an impact in the runtime of the thermal simulation. As a result, fewer generations of the evolutionary floorplanner are executed. Therefore, it might be possible to achieve thermally and performance optimized configurations with 3D-ICEc if a longer number of iterations of the optimization process are run. Thus, we compare the configurations obtained with the two studied thermal models with a fixed number of generations.

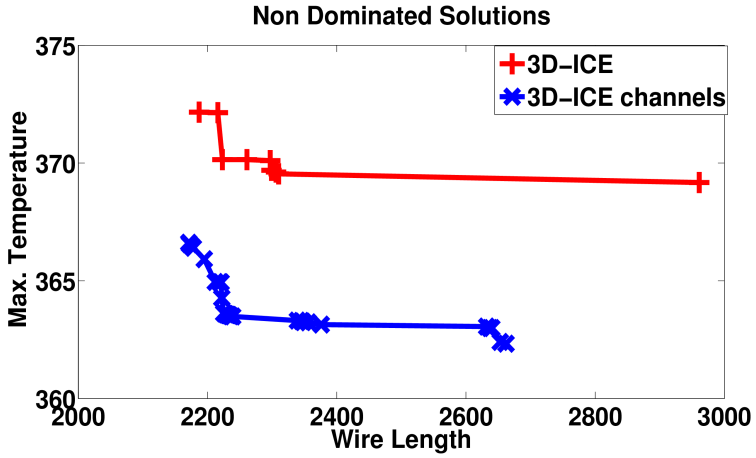
Fixed Generations

Figure 6.8 shows the nondominated solutions retrieved from the 10 replicas of the experiments with 500 generations. As in the previous figures, temperatures have been obtained with 3D-ICE including the corresponding cooling layers.

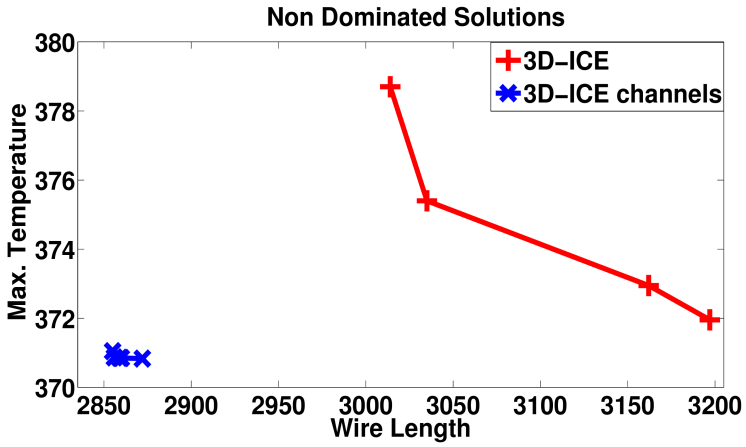
In the 48 cores scenario, the solutions obtained with 3D-ICEc reach a lower temperature while they exhibit a similar wire length. For example, the solution obtained with 3D-ICE presenting the shortest wire length reaches a temperature of $372.2K$ while a solution obtained with 3D-ICEc exhibiting a similar performance presents a maximum chip temperature of $366.4K$. Thus, the front of nondominated solutions obtained with 3D-ICEc dominates the set of configurations retrieved with the standard 3D-ICE.

The suitability of the 3D-ICEc model is confirmed in the 64 cores scenario. The configurations retrieved with the referred model still present temperature up to $7.8K$ lower. Moreover, the performance difference is clearly increased. In fact, the so-obtained configurations present a performance improvement in the range of 5.3% to 10.7%.

Therefore, it is clearly necessary to include the microfluidic cooling in the optimization process as it leads to simultaneously reduce temperature and wire length of the studied architectures.



(a) 48 cores scenario



(b) 64 cores scenario

Figure 6.8: Optimized solutions found with and without including microchannels in the optimization process in the 48, and 64 scenarios considering a fixed number of generations

6.3.3 Conclusions

In this section, we have integrated two accurate thermal models in the Direct Mapping floorplanner. The first model is the standard 3D-ICE while the second simulates the inclusion of intermediate cooling layers in the 3D stack. The added layers contain micro-channels placed in a homogeneous manner. A liquid coolant runs through these channels at a fixed flow rate. These layers allow to relax the severe thermal constraints imposed by large 3D MPSoCs. As a result, there are extra optimization opportunities that must be exploited by thermal-aware floorplanners.

The proposed Direct Mapping floorplanner is capable of exploiting these opportunities. In fact, considering the inclusion of microchannels in the optimization process simultaneously reduces temperature and wire length of the targeted large 3D MPSoCs.

The integration of the 3D-ICEc model leads to a long optimization time that must be reduced. However, we have already studied in Chapter 5 the possibility of training a neural network to mimic the results of 3D-ICE (NNTM model). Such procedure could be reused here to reproduce the results of 3D-ICEc, reducing the runtime of the thermal simulation of the optimized architectures with liquid cooling. The NNTM model would still be suitable for GPU implementation, ensuring the adequacy of the proposed methodology.

Chapter 7

Conclusions and Future Work

The conclusions presented in previous chapters are summarized in the following. We also discuss possible improvements and potential future works. Finally, we give the references of the publications resulting from the presented research and of the projects and grants with which this work has been funded.

7.1 Conclusions

In this thesis, we have faced a relevant real-world problem, the thermal aware floorplanning of large 3D MPSoCS. The silicon industry is moving towards chips with a large number of integrated processors. However, these platforms are limited by the elevated number and high latency of bus transactions. 3D integration is accepted as a viable way to reduce the overall wire length of the chip while increasing the bandwidth between vertical layers by means of Through-Silicon Vias. Therefore, these architectures are expected to provide the demanded performance increase in the coming years.

It has been demonstrated that the temperatures reached by large 3D MPSoCs remain in acceptable levels. Thermal-aware floorplanning alone allows to significantly reduce chip temperature. However, the prohibitive peak temperatures appearing in the targeted architectures make necessary the incorporation of additional cooling techniques. We have simulated the inclusion of microfluidic channels disposed in a homogeneous manner between the active layers of the 3D chip. A liquid coolant such

as water runs along these channels at a fixed flow rate. The considered technique allows to dramatically reduce chip temperature, achieving feasible temperatures. The power consumption required by such cooling technique is expected to remain in admissible values, even though the specific values will depend on the number of channels and the specific characteristics of the involved micropump(s).

We have showed the necessity for new techniques capable of dealing with the severe thermal constraints imposed by large 3D MPSoCs. Relevant floorplanning proposals such as Combined Bucket and 2D Array, Double-Tree and Sequence, Sequence Pair, and Generalized Polish Expression have been employed to perform the optimization of three platforms based on the Niagara architecture. A comparative study has showed that the cited techniques do not reach feasible solutions when an initial configuration of the architecture is not provided. Moreover, there is a need for multi-objective techniques as the studied chips present a tradeoff between temperature and wire length. In fact, floorplanning tools must provide a wide range of solutions presenting a good tradeoff between the referred conflicting objectives so that chip manufacturers can select the best configuration according to their own criteria.

Multi-Objective Evolutionary Algorithms have been employed to deal with the above mentioned tradeoff between performance and temperature. These algorithms extend Evolutionary Algorithms with the incorporation of the concept of Pareto optimality in the optimization process. MOEAs represent a suitable tool to face complex multi-objective problems, obtaining a set of solutions simultaneously optimizing two or more conflicting objectives in a reduced time.

New methodologies have been proposed leading to a temperature reduction and performance enhancement of the targeted architectures. A previous proposal, namely the Multi-objective Floorplanning Algorithm (MFA) has been improved in several ways:

1) First, we have accelerated the optimization process with a parallel implementation. In particular, a multi-threaded Master-worker model has been adopted producing a relevant speedup in the architectural exploration task.

2) We have introduced dynamic knowledge in the floorplanning process, a typically static optimization technique. To this end, we have considered power profiles retrieved from architectural simulations. However, including such knowledge in the optimization phase has revealed to be a hard task. In fact, the added information has been reduced to several metrics since the considered MOEAs do not perform well when an elevated number of objectives is targeted.

3) We have evaluated the impact of integrating different thermal models in the optimization process as a tradeoff between runtime of the thermal model and accuracy has typically been assumed. We have considered an exact model, namely 3D-ICE and an approximated but fast thermal model. Additionally, we have incorporated to our floorplanner the recently proposed Neural Network Thermal Model. Such model is trained to mimic the outputs of 3D-ICE and had never been used before in a floorplanning context. Moreover, a GPGPU version of such model has been employed, eliminating the referred tradeoff between runtime and accuracy. In fact, the trained neural network is faster than the approximated thermal model and performs accurate simulations. Thus, with a fixed optimization time, the results obtained with the NNTM outperform the optimized configurations achieved with other thermal models.

A major contribution of this thesis is the proposal of Direct Mapping, a new representation designed for fixed-outline floorplanning problems that allows to perform architectural exploration tasks in an efficient manner. Direct Mapping exploits the modular design enabled by 3D integration as the components of a given architecture are treated as IP blocks. Moreover, it is totally compatible with existing tools such as thermal simulators that split the surface of a chip in rectangular thermal cells. Therefore, the proposed method perfectly fits in a toolchain covering different steps of the architectural design process such as architectural exploration, optimization, and validation among others. This representation allows for a direct mapping of the individuals into configurations of the architecture avoiding the costly decoding step of optimization algorithms. Furthermore, it eliminates heuristics that might limit the exploration space and cause premature convergence problems.

A Multi-Objective Evolutionary Algorithm based on NSGA-II has been designed to take advantage of the proposed representation. The new floorplanning proposal has been validated and compared against state-of-the-art floorplanners. The performed experiments show that the use of Direct Mapping together with the proposed problem-specific operators leads to optimal results, outperforming all the analyzed techniques.

It has been shown that the inclusion of microfluidic channels in intermediate layers of the 3D chip greatly impacts the thermal profile of the targeted architectures. We have demonstrated that it is necessary to consider this cooling effect during the optimization process to achieve an optimal thermal behavior and performance. In fact, floorplanners must benefit from the extra optimization opportunities provided by the relaxation of the thermal restrictions.

In summary, we have proposed bioinspired heuristics that successfully tackle the optimization of large 3D MPSoCs. We have outperformed previous proposals by means of introducing problem-specific knowledge to the floorplanning process in the form of power profiles, suitable thermal models, and a new representation. Moreover, parallel approaches have been implemented to provide an efficient tool capable of performing the architectural exploration of large 3D MPSoCs.

7.2 Future Work

This work has shown that large 3D platforms are feasible in terms of temperature and provide performance improvement due to a reduced wire length. There are two main ways of extending the techniques presented in this thesis. The first way consists in considering further architectural constraints in the optimization process. Second, some of the proposed techniques have the potential to be improved.

Regarding the architectural constraints, even though 3D integration technologies are continuously evolving, it is possible to incorporate more features such as a realistic interprocessor communication. For instance, the architectures based on the Niagara platform studied in this work contain a set of crossbars for communication. However, there is a trend towards the implementation of a Network-on-Chip (NoC) when the number of integrated processors is high. In such designs, a packet-based communication is enabled with a procedure similar to the Internet Protocol. Packets pass through switches that must be floorplanned appropriately. Moreover, the number of total switches must be minimized to reduce power consumption. The design of these embedded networks alone represents a challenging problem and is becoming a relevant research topic [15], [116], [11], and [61]. Just as accurate thermal simulators have been employed in this work, it is necessary to consider realistic communication models. This way, the performance improvement provided by large 3D MPSoCs will be obtained in a more accurate manner.

Another architectural issue not addressed in this work is the TSV placement. Note that this problem has already been approached within the research group by David Cuesta *et al.* [38] and [36], and that the cited works are compatible with the methodologies presented in this thesis. Broadly speaking, the placement of TSVs can be performed in two ways. The first approach consists in a two phase procedure in which the placement of TSVs is performed after the placement of the components of the architectures or vice versa. The other possibility is the adoption of a "TSV-as-cell" approach, in which both components and vias are modeled as blocks that must be placed in the same floorplanning step. The proposed Direct Mapping representation is compatible with both alternatives although the latter would result

in a significant area overhead since the areas of the considered components must be at least equal to the size of the fixed cell ($300\mu m$ in the performed experiments).

The employed cooling technique also presents room for improvement as explained in [39]. We have simulated the integration of microfluidic channels with fixed flow rate to demonstrate the feasibility of the studied large 3D architectures. However, depending on the specific pump characteristics, it could be possible to allow a variant flow rate so that power consumption is minimized while temperature remains in acceptable levels. Moreover, the floorplanning and number of channels can be considered from early stages, allowing new optimization opportunities.

With respect to the proposed techniques, we have successfully employed MOEAs, obtaining configurations presenting a good tradeoff between temperature and wire length. It has also been demonstrated that the retrieved configurations outperform the results obtained with other well-known floorplanning techniques. Moreover, other proposals based on Simulated Annealing such as Combined Bucket and 2D Array (CBA) are not suitable for architectural exploration of large 3D MPSoCs as they require an initial feasible floorplan. In fact, these techniques carry out a fine tuning of an initial configuration mainly by means of local search operators. It would be therefore beneficial to combine the exploration capability of the studied MOEAs with the exploitation performed by methods such as CBA. A two step optimization process could easily be implemented in which, in the first place, a front of nondominated solutions would be obtained with a MOEA. Later, CBA would be applied to the most promising configurations of the retrieved front.

Another approach could be adopted to take advantage from dynamic information such as the considered power profiles. Note that, in the methodology proposed in Section 5.3, several metrics were extracted from the power traces to guide the thermal optimization. In fact, candidate architectures cannot be simulated with all the retrieved execution traces due to optimization time constraints. However, faster simulators would allow to analyze the thermal behavior of the different floorplans while executing real-world applications. This, in turn, would allow to study the evolution of metrics such as the maximum thermal gradient over time, directly related

to the Average Chip Lifetime or Mean Time Between Failures. Therefore, exploiting the dynamic information retrieved from the execution traces can be used to guide 3D MPSoC design towards more reliable and long-lasting architectures.

Also, in the absence of quicker simulation tools, a more complex but faster acquisition of knowledge from the execution traces is possible. In a recent work [157], Wu *et al.* propose a load-aware Dynamic Voltage and Frequency Scaling technique. The workloads are also retrieved from simulations but, in this case, the obtained traces are transformed to the frequency domain. With this elegant approach, the amount of information managed by the algorithm is significantly reduced and the dynamic information can be fully exploited. The management of this dynamic information can be used to exploit low-power modes in the design stage of the optimization process. This way, voltage islands techniques can be applied (see [106] and [70]), resulting in a better energy efficiency which leads to lower temperatures.

7.3 Discussion

This works proposes bioinspired heuristics for the floorplanning of large 3D MPSoCs. It is relevant to note the difference between heuristic and architectural validation as these two worlds somehow collide in some aspects of this work. The appropriate performance of heuristics must be statistically validated by means of a reproducible experimental setup. On the other hand, in an architectural design context, obtaining a single solution satisfying existing constraints such as temperature, feasibility and wire length might suffice. However, when working with Evolutionary Algorithms, it is not trivial to deal with the referred constraints. In fact, we have detected that imposing hard constraints usually has a negative effect in the search dynamics. For instance, it would be reasonable to set maximum temperature or wire length thresholds during the search process to discard bad performing candidate solutions. However, we have empirically noticed that imposing such hard constraints leads to a partial exploration of the solution space, producing suboptimal solutions.

Broadly speaking, the goal is to provide best possible conditions for the heuristic search process. Such optimal conditions are sometimes counter-intuitive. To illustrate this idea, the reader is referred to Section 5.3, where we discuss the suitability of several thermal metrics as guiding cost function of the optimization process. It is shown that peak temperatures are reached when all the components exhibit a maximum power consumption. However, it is not beneficial to consider these maximum power consumptions in the optimization process to reduce the final peak temperatures. In fact, a relaxation of the thermal restrictions during the floorplanning process results in a better thermal optimization.

Another choice that deeply impacts the dynamics of the search process is the chosen representation. In fact, different representations lead to different solution spaces. These solution spaces might not only present a different cardinality but also a different distribution of global and local optima. Thus, some representations will lead to fitness landscapes better adapted for heuristic search than others. A brief introduction to Fitness Landscape Analysis is given in Section 4.1.4 in the context of mono-objective optimization. However, there is a lot of research needed

to better understand the information retrieved from such analyses. The final goal of fitness landscape analysis should be to find optimal algorithms for a given landscape. However, this step is typically missing in most of the works dealing with the subject.

Such analysis would be extremely useful in the context of complex problem solving in which typically heuristics are employed. The main challenge remains the huge cardinality of the studied solution spaces. Sampling techniques such as random walks are usually employed to assess properties describing the analyzed landscapes (ruggedness, deception, multimodality etc..). Some promising works have revealed that the distribution of global and local optima in the solution space are far from being random [42]. However, the presented results are not necessarily universal and the proposed technique is hardly scalable as it is based on exhaustive procedures. However, the current development of data-mining techniques for BigData scenarios might provide a greater knowledge of the distribution of the searched optima. All these difficulties are incremented when multi-objective problems are analyzed. In such case, it is possible to analyze the relation between solutions or between fronts (sets) of solutions as explained in [151], a promising work by S. Verel.

In brief, it can be said that researchers are only starting to discover how the solutions of a given problem relate to each other, how to classify the different problems according to Fitness Landscape Analysis techniques, and how to design efficient algorithms for the analyzed landscapes.

7.4 Publications

We now give the references of the publications resulting from this work:

Journal publications:

- Arnaldo, I.; Risco-Martín, J.L.; Ayala, J.L.; Hidalgo, J.I.: *Power profiling-guided floorplanner for 3D multiprocessor systems-on-chip,* Circuits, Devices & Systems, IET, vol.6, no.5, pp.322-329, Sept. 2012.

- I. Arnaldo, J. L. Risco-Martín, J.L. Ayala, J.M. Colmenar, and A. Cuesta-Infante: *Boosting the 3D thermal-aware floorplanning problem through a master-worker parallel MOEA.* Concurrency and Computation: Practice and Experience, Wiley, 2013.

- Arnaldo, I., Contreras, I., Millán-Ruiz, D., Hidalgo, J. I. and Krasnogor, N. *Matching island topologies to problem structure in parallel evolutionary algorithms.* International Journal of Soft Computing: Special Issue on Bio-inspired Algorithms with Structured Populations, Springer, 2013.

Workshops and congress publications:

- I. Arnaldo, J. L. Risco-Martín, J. L. Ayala, and J. I. Hidalgo. *Power Profiling-Guided Floorplanner for Thermal Optimization in 3D Multiprocessor Architectures*, in Integrated Circuit and System Design. Power and Timing Modeling, Optimization, and Simulation, ser. Lecture Notes in Computer Science, J. Ayala, B. García-Cámara, M. Prieto, M. Ruggiero, and G. Sicard, Eds. Springer Berlin / Heidelberg, 2011, vol. 6951, pp. 11-21.

- Arnaldo, I., Risco-Martín, J., Ayala, J., Colmenar, J. M., Cuesta-Infante, A. and Hidalgo, J. I. *Parallel MOEA Implementation for Fast Evaluation of the 3D Thermal-Aware Floorplanning Probem.* In 4th Workshop on Parallel Architectures and Bioinspired Algorithms, WPABA. 2011.

- Arnaldo, I., Risco-Martín, J., Ayala, J., Colmenar, J. M., Cuesta-Infante, A. and Hidalgo, J. I. *AEMO Paralelo para Resolver el Problema del Floorplanning Térmico en 3 Dimensiones.* In VIII Congreso Español sobre Metaheurísticas, Algoritmos Evolutivos y Bioinspirados, MAEB. 2011.

- Arnaldo, I., Vicenzi, A., Ayala, J., Risco-Martin, J., Hidalgo, J. I., Ruggiero, M. y Atienza, D. *Fast and Scalable Temperature-driven Floorplan Design in 3D MPSoCs.* En 13th IEEE Latin American Test Workshop (LATW). 2011.

Master thesis (first Phd year):

- Arnaldo, I. *Evolutionary Approaches to Solve the 3D Thermal-Aware Floorplanning Problem using Heterogeneous Processors.* Proyecto Fin de Máster, Universidad Complutense de Madrid, 2011.

7.5 Research Projects and Grants

This work has been partially supported by the following projects and grants:

- Project: *AMBU: Arquitectura de servicios de supercomputación en la nube*, funded by the Spanish Government, Ministerio de Industria, Turismo y Comercio, Plan Nacional de Investigación Científica, Desarrollo e Innovación Tecnológica 2008-2011. Avanza Competitividad I+D+I: TSI-020100-2010-962

- Project: *IYELMO: Plataforma de servicios en la nube para operaciones en mercados financieros*, funded by Spanish Government, Ministerio de Ciencia e Innovación. INNPACTO-IPT-2011-1198-430000

- Mobility Grant *Orden ECD /3628/2011, de 26 de diciembre*: A three-month stay in the Nottingham University has been funded with a mobility grant from the Spanish Government, Dirección General de Política Universitaria, Ministerio de Educación, Cultura y Deporte.

Bibliography

[1] S.N. Adya and I.L. Markov. Fixed-outline floorplanning: enabling hierarchical design. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 11(6):1120 –1135, dec. 2003.

[2] S.F. Al-Sarawi, D. Abbott, and P.D. Franzon. A review of 3-D packaging technology. *Components, Packaging, and Manufacturing Technology, Part B: Advanced Packaging, IEEE Transactions on*, 21(1):2 –14, feb 1998.

[3] E. Alba and M. Tomassini. Parallelism and evolutionary algorithms. *Evolutionary Computation, IEEE Transactions on*, 6(5):443 – 462, oct 2002.

[4] Enrique Alba and José M. Troya. Influence of the Migration Policy in Parallel Distributed GAs with Structured and Panmictic Populations. *Applied Intelligence*, 12(3):163–181, May 2000.

[5] Lourdes Araujo and Juan Julián Merelo Guervós. Diversity through multiculturality: Assessing migrant choice policies in an island model. *IEEE Trans. Evolutionary Computation*, 15(4):456–469, 2011.

[6] ARM. www.arm.com/products/processors/cortex-a/cortex-a9.php, 2011.

[7] Ignacio Arnaldo. Evolutionary approaches to solve the 3D thermal-aware floorplanning problem using heterogeneous processors. Master's thesis, Complutense University, Madrid, 2011.

[8] Ignacio Arnaldo, Iván Contreras, David Millán-Ruiz, J. Ignacio Hidalgo, and Natalio Krasnogor. Matching island topologies to problem structure in parallel evolutionary algorithms. *Soft Computing: Special issue on structured populations*, 2013.

[9] D. Atienza. Thermal-aware design of 3D ICs with inter-tier liquid cooling. In *Electron Devices Meeting (IEDM), 2010 IEEE International*, page 17.2.1, dec. 2010.

[10] David Atienza, Pablo G. Del Valle, Giacomo Paci, Francesco Poletti, Luca Benini, Giovanni De Micheli, Jose M. Mendias, and Roman Hermida. HW-SW emulation framework for temperature-aware design in MPSoCs. *ACM Trans. Des. Autom. Electron. Syst.*, 12(3):26:1–26:26, May 2008.

[11] J. L. Ayala, M. Lopez-Vallejo, D. Bertozzi, and L. Benini. *SoC Communication Architectures: From Interconnection Buses to Packet-Switched NoCs*, pages 1401–1429. Industrial Information Technology. CRC Press, 2009.

[12] S. Bandyopadhyay, S. Saha, U. Maulik, and K. Deb. A Simulated Annealing-Based Multiobjective Optimization Algorithm: AMOSA. *IEEE Transactions on Evolutionary Computation*, 12(3):269–283, June 2008.

[13] IBM-PLACE Benchmarks. http://er.cs.ucla.edu/benchmarks/ibm-place, 2013.

[14] Gerardo Beni and Jing Wang. Swarm intelligence in cellular robotic systems. In Paolo Dario, Giulio Sandini, and Patrick Aebischer, editors, *Robots and Biological Systems: Towards a New Bionics?*, volume 102 of *NATO ASI Series*, pages 703–712. Springer Berlin Heidelberg, 1993.

[15] L. Benini and G. De Micheli. Networks on chips: a new SoC paradigm. *Computer*, 35(1):70 –78, jan 2002.

[16] Kerry Bernstein. *Three-Dimensional Integrated Circuit Design*, chapter Introduction, pages 1–13. Springer Publishing Company, Incorporated, 2009.

[17] Johan Berntsson and Maolin Tang. A slicing structure representation for the multi-layer floorplan layout problem. In *EvoWorkshops*, pages 188–197, 2004.

[18] David Brooks, Robert P Dick, Russ Joseph, and Li Shang. Power, thermal, and reliability modeling in nanometer-scale microprocessors. *IEEE Micro*, 27(3):49–62, 2007.

[19] J.A. Burns, B.F. Aull, C.K. Chen, Chang-Lee Chen, C.L. Keast, J.M. Knecht, V. Suntharalingam, K. Warner, P.W. Wyatt, and D.-R.W. Yost. A wafer-scale 3-D circuit integration technology. *Electron Devices, IEEE Transactions on*, 53(10):2507 –2516, oct. 2006.

[20] Erick Cantú-Paz. Migration policies and takeover times in genetic algorithms. In *GECCO*, page 775, 1999.

[21] Erick Cantu-Paz. Topologies, migration rates, and multi-population parallel genetic algorithms, 1999.

[22] Song Chen and T. Yoshimura. Fixed-outline floorplanning: Block-position enumeration and a new method for calculating area costs. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 27(5):858 –871, may 2008.

[23] Tung-Chieh Chen, Yao-Wen Chang, and Shyh-Chang Lin. A new multilevel framework for large-scale interconnect-driven floorplanning. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 27(2):286 –294, feb. 2008.

[24] Ting-Yen Chiang, S.J. Souri, Chi On Chui, and K.C. Saraswat. Thermal analysis of heterogeneous 3D ICs with various integration scenarios. In *Electron Devices Meeting, 2001. IEDM Technical Digest. International*, pages 31.2.1 –31.2.4, 2001.

[25] Carlos A. Coello Coello, Gary B. Lamont, and David A. van Veldhuizen. *Evolutionary Algorithms for Solving Multi-Objective Problems*. Kluwer, 2002.

[26] Carlos A. Coello Coello, Gregorio Toscano Pulido, and Efrén Mezura Montes. Current and Future Research Trends in Evolutionary Multiobjective Optimization, 2005.

[27] J.P. Cohoon, S.U. Hegde, W.N. Martin, and D. Richards. Floorplan design using distributed genetic algorithms. In *Computer-Aided Design, 1988. ICCAD-88. Digest of Technical Papers., IEEE International Conference on*, pages 452 –455, nov 1988.

[28] Taiwan Semiconductor Manufacturing Company. www.tsmc.com/english/dedicatedFoundry/technology/90nm.htm, 2013.

[29] J. Cong and Guojie Luo. A multilevel analytical placement for 3D ICs. In *Design Automation Conference, 2009. ASP-DAC 2009. Asia and South Pacific*, pages 361–366, 2009.

[30] J. Cong, Guojie Luo, Jie Wei, and Yan Zhang. Thermal-aware 3d ic placement via transformation. In *Design Automation Conference, 2007. ASP-DAC '07. Asia and South Pacific*, pages 780–785, 2007.

[31] J. Cong, Jie Wei, and Yan Zhang. A thermal-driven floorplanning algorithm for 3D ICs. In *Proceedings of the 2004 IEEE/ACM International conference on Computer-aided design*, ICCAD '04, pages 306–313, Washington, DC, USA, 2004. IEEE Computer Society.

[32] David W. Corne, Joshua D. Knowles, and Martin J. Oates. The pareto envelope-based selection algorithm for multiobjective optimization. In *Proceedings of the Parallel Problem Solving from Nature VI Conference*, pages 839–848. Springer, 2000.

[33] Oracle Corporation. http://www.oracle.com/us/products/servers-storage/servers/sparc-enterprise/t-series/overview/index.html, 2010.

[34] A.K. Coskun, J.L. Ayala, D. Atienza, T.S. Rosing, and Y. Leblebici. Dynamic thermal management in 3D multicore architectures. In *Design, Automation Test in Europe Conference Exhibition, 2009. DATE '09.*, pages 1410 –1415, april 2009.

[35] Nichael Lynn Cramer. A representation for the adaptive generation of simple sequential programs. In *Proceedings of the 1st International Conference on Genetic Algorithms*, pages 183–187, Hillsdale, NJ, USA, 1985. L. Erlbaum Associates Inc.

[36] D. Cuesta, J. L. Risco-Martin, and J. L. Ayala. 3D thermal-aware floorplanner using a MILP approximation. *Microprocessors and Microsystems*, 36:344–354, 2012.

[37] D. Cuesta, J.L. Risco-Martin, J.L. Ayala, and J I. Hidalgo. A combination of evolutionary algorithm and mathematical programming for the 3D thermal-aware floorplanning problem. In *13th annual conference on Genetic and evolutionary computation (GECCO 2011)*, pages 1731–1738. ACM Press, ACM Press, 07/2011 2011.

[38] David Cuesta, José L. Risco-Martín, José L. Ayala, and J. Ignacio Hidalgo. 3D thermal-aware floorplanner using a MOEA approximation. *Integration, the VLSI Journal*, 2012.

[39] David Cuesta, José L. Risco-Martín, José L. Ayala, and J. Ignacio Hidalgo. Thermal-Aware Floorplanner for 3D IC, including TSVs, Liquid Microchannels and Thermal Domains Optimization. *submitted to ACM Transactions on Design Automation of Electronic Systems*, 2013.

[40] Alfredo Cuesta-Infante, J. Manuel Colmenar, Zorana Bankovic, José L. Risco-Martín, Marina Zapater, J. Ignacio Hidalgo, José L. Ayala, and José M. Moya. Comparative study of meta-heuristic 3D floorplanning algorithms. *Applied Soft Computing*, In Press, 2013.

[41] Bing Dang, M.S. Bakir, D.C. Sekar, C.R. King, and J.D. Meindl. Integrated Microfluidic Cooling and Interconnects for 2D and 3D Chips. *Advanced Packaging, IEEE Transactions on*, 33(1):79 –87, feb. 2010.

[42] Fabio Daolio, Marco Tomassini, Sébastien Vérel, and Gabriela Ochoa. Communities of minima in local optima networks of combinatorial spaces. *Physica A: Statistical Mechanics and its Applications*, 390(9):1684 – 1694, 2011.

[43] C. Darwin. *On the Origin of Species by Means of Natural Selection, or the Preservation of Favoured Races in the Struggle for Life*. J. Murray, sixth edition, 1872.

[44] W.R. Davis, J. Wilson, S. Mick, J. Xu, H. Hua, C. Mineo, A.M. Sule, M. Steer, and P.D. Franzon. Demystifying 3d ics: the pros and cons of going vertical. *Design Test of Computers, IEEE*, 22(6):498 – 510, nov.-dec. 2005.

[45] Richard Dawkins. *The Selfish Gene*. Oxford University Press, September 1990.

[46] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *Evolutionary Computation, IEEE Transactions on*, 6(2):182 –197, apr 2002.

[47] M. Dorigo. *Optimization, Learning and Natural Algorithms*. PhD thesis, Politecnico di Milano, Italy, 1992.

[48] B. Dorronsoro and P. Bouvry. Improving classical and decentralized differential evolution with new mutation operator and population topologies. *Evolutionary Computation, IEEE Transactions on*, 15(1):67 –98, feb. 2011.

[49] A. Drakidis, R.J. Mack, and R.E. Massara. Packing-based VLSI module placement using genetic algorithm with sequence-pair representation. *Circuits, Devices and Systems, IEE Proceedings* -, 153(6):545 –551, dec. 2006.

[50] F.Y. Edgeworth. *Mathematical Psychics*. History of Economic Thought Books. McMaster University Archive for the History of Economic Thought, Hamilton, ON, CA, 1881.

[51] M. Ekpanyapong et al. Thermal-aware 3D Microarchitectural Floorplanning. Technical report, Georgia Institute of Technology Center, 2004.

[52] P. Emma and E. Kursun. Opportunities and Challenges for 3D Systems and Their Design. *Design Test of Computers, IEEE*, 26(5):6 –14, sept.-oct. 2009.

[53] Mark Erickson, Alex Mayer, and Jeffrey Horn. The Niched Pareto Genetic Algorithm 2 Applied to the Design of Groundwater Remediation Systems. In *Proceedings of the First International Conference on Evolutionary Multi-Criterion Optimization*, EMO '01, pages 681–695, London, UK, UK, 2001. Springer-Verlag.

[54] Lawrence J. Fogel, Peter J. Angeline, and David B. Fogel. An Evolutionary Programming Approach to Self-Adaptation on Finite State Machines. In *Proceedings of the fourth annual Conference on Evolutionary Programming*, pages 355–365. MIT Press, 1995.

[55] Carlos M. Fonseca and Peter J. Fleming. Genetic Algorithms for Multiobjective Optimization: Formulation, Discussion and Generalization. In *Genetic*

Algorithms: Proceedings of the Fifth International Conference, pages 416–423. Morgan Kaufmann, 1993.

[56] International Technology Roadmap for Semiconductors (ITRS). Design. http://www.itrs.net/Links/2011ITRS/2011Chapters/2011Design.pdf, 2011.

[57] Kunihiro Fujiyoshi, Hidenori Kawai, and Keisuke Ishihara. A tree based novel representation for 3D-block packing. *Trans. Comp.-Aided Des. Integ. Cir. Sys.*, 28(5):759–764, May 2009.

[58] M. R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.

[59] Mario Giacobini, Mike Preuss, and Marco Tomassini. Effects of Scale-Free and Small-World Topologies on Binary Coded Self-adaptive CEA. *Evolutionary Computation in Combinatorial Optimization*, pages 86–98, 2006.

[60] Mario Giacobini, Marco Tomassini, and Andrea Tettamanzi. Takeover time curves in random and small-world structured populations. In *Proceedings of the 2005 conference on Genetic and evolutionary computation*, GECCO '05, pages 1333–1340, New York, NY, USA, 2005. ACM.

[61] Francisco Gilabert, Davide Bertozzi, Luca Benini, and Giovanni De Micheli. *Networks-on-Chip: An Interconnect Fabric for Multiprocessor Systems-on-Chip*, pages 15/1 – 15/29. Industrial Information Technology. CRC Press, 2009.

[62] Fred Glover. Future paths for integer programming and links to artificial intelligence. *Comput. Oper. Res.*, 13(5):533–549, May 1986.

[63] M.P. Gupta, Minki Cho, S. Mukhopadhyay, and S. Kumar. Thermal mangament of multicore processors using power multiplexing. In *Thermal and Thermomechanical Phenomena in Electronic Systems (ITherm), 2010 12th IEEE Intersociety Conference on*, pages 1 –7, june 2010.

[64] M. Healy, M. Vittes, M. Ekpanyapong, et al. Multiobjective Microarchitectural Floorplanning for 2-D and 3-D ICs. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 26(1):38 –52, jan. 2007.

[65] J. H. Holland. *Adaptation in Natural and Artificial Systems*. The University of Michigan Press, 1975.

[66] J. Horn, N. Nafpliotis, and D.E. Goldberg. A niched Pareto genetic algorithm for multiobjective optimization. In *Evolutionary Computation, 1994. IEEE World Congress on Computational Intelligence., Proceedings of the First IEEE Conference on*, pages 82 –87 vol.1, jun 1994.

[67] HPlabs. www.hpl.hp.com/research/cacti/, 2011.

[68] Jian Jun Hu and E.D. Goodman. The hierarchical fair competition (HFC) model for parallel evolutionary algorithms. In *Evolutionary Computation, 2002. CEC '02. Proceedings of the 2002 Congress on*, volume 1, pages 49 –54, may 2002.

[69] W.-L. Hung, Y. Xie, N. Vijaykrishnan, C. Addo-Quaye, T. Theocharides, and M.J. Irwin. Thermal-aware floorplanning using genetic algorithms. In *Quality of Electronic Design, 2005. ISQED 2005. Sixth International Symposium on*, pages 634 – 639, march 2005.

[70] W.L. Hung, G.M. Link, Yuan Xie, N. Vijaykrishnan, N. Dhanwadaf, and J. Conner. Temperature-aware voltage islands architecting in system-on-chip design. In *Computer Design: VLSI in Computers and Processors, 2005. ICCD 2005. Proceedings. 2005 IEEE International Conference on*, pages 689 – 694, oct. 2005.

[71] Chang-Gyu Hwang. New paradigms in the silicon industry. In *Electron Devices Meeting, 2006. IEDM '06. International*, pages 1 –8, dec. 2006.

[72] IBM. Complex SoC Design, China Design Center IBM. http://www.ibm.com/systems/support/tools/estimator/energy, 2009.

[73] IBM. http://www.ibm.com/systems/support/tools/estimator/energy, 2011.

[74] C. Igel and M. Kreutz. Using fitness distributions to improve the evolution of learning structures. In *Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on*, volume 3, pages 3 vol. (xxxvii+2348), 1999.

[75] Sungjun Im and K. Banerjee. Full chip thermal analysis of planar (2-D) and vertically integrated (3-D) high performance ICs. In *Electron Devices Meeting, 2000. IEDM Technical Digest. International*, pages 727 –730, 2000.

[76] Imperas. www.ovpworld.org, 2011.

[77] Syed Muhammad Zeeshan Iqbal, Yuchen Liang, and Hakan Grahn. ParMiBench - an open-source benchmark for embedded multiprocessor systems. *CAL*, 9(2):45 –48, feb. 2010.

[78] P. Jacob, O. Erdogan, A. Zia, P.M. Belemjian, R.P. Kraft, and J.F. McDonald. Predicting the performance of a 3D processor-memory chip stack. *Design Test of Computers, IEEE*, 22(6):540 – 547, nov.-dec. 2005.

[79] S. Kawamura, N. Sasaki, T. Iwai, M. Nakano, and M. Takagi. Three-dimensional CMOS IC's Fabricated by using beam recrystallization. *Electron Device Letters, IEEE*, 4(10):366 – 368, oct 1983.

[80] J. Kennedy and R. Eberhart. Particle swarm optimization. In *Neural Networks, 1995. Proceedings., IEEE International Conference on*, volume 4, pages 1942–1948 vol.4. IEEE, November 1995.

[81] Scott Kirkpatrick. Optimization by simulated annealing: Quantitative studies. *Journal of Statistical Physics*, 34:975–986, 1984.

[82] J.U. Knickerbocker, C.S. Patel, P.S. Andry, C.K. Tsang, L.P. Buchwalter, E.J. Sprogis, Hua Gan, R.R. Horton, R.J. Polastre, S.L. Wright, and J.M. Cotte. 3-D Silicon Integration and Silicon Packaging Technology Using Silicon Through-Vias. *Solid-State Circuits, IEEE Journal of*, 41(8):1718 –1725, aug. 2006.

[83] Joshua Knowles and David Corne. Approximating the non-dominated front using the Pareto Archived Evolution Strategy. *Evolutionary Computation*, 8:149–172, 1999.

[84] Seiichi Koakutsu and Hironori Hirata. Genetic simulated annealing for floorplan design. In Jacques Henry and Jean-Pierre Yvon, editors, *System Mod-*

elling and Optimization, volume 197 of *Lecture Notes in Control and Information Sciences*, pages 268–277. Springer Berlin Heidelberg, 1994.

[85] M. Koyanagi, T. Fukushima, and T. Tanaka. High-Density Through Silicon Vias for 3-D LSIs. *Proceedings of the IEEE*, 97(1):49 –59, jan. 2009.

[86] M. Koyanagi, H. Kurino, Kang Wook Lee, K. Sakuma, N. Miyakawa, and H. Itani. Future system-on-silicon LSI chips. *Micro, IEEE*, 18(4):17 –22, jul/aug 1998.

[87] John R. Koza. *Genetic programming: on the programming of computers by means of natural selection*. MIT Press, Cambridge, MA, USA, 1992.

[88] Natalio Krasnogor. An unorthodox introduction to memetic algorithms. *SIGEVOlution*, 3(4):6–15, December 2008.

[89] Natalio Krasnogor and Steven Gustafson. A study on the use of "self-generation" in memetic algorithms. *Natural Computing*, 3:53–76, 2004. 10.1023/B:NACO.0000023419.83147.67.

[90] Ding-Ming Kwai. Homogeneous integration for 3D IC with TSV. In *Proceedings of the 2010 Asia and South Pacific Design Automation Conference*, ASPDAC '10, pages 546–547, Piscataway, NJ, USA, 2010. IEEE Press.

[91] Ding-Ming Kwai, Yung-Fa Chou, and Cheng-Wen Wu. Is 3D integration the way out of the crossroads? In *Solid State Circuits Conference (A-SSCC), 2010 IEEE Asian*, pages 1 –4, nov. 2010.

[92] J.H. Lau. Evolution and outlook of TSV and 3D IC/Si integration. In *Electronics Packaging Technology Conference (EPTC), 2010 12th*, pages 560 –570, dec. 2010.

[93] Changgu Lee and L.G. Fréchette. A Silicon Microturbopump for a Rankine-Cycle Power Generation Microsystem. Part I: Component and System Design. *Microelectromechanical Systems, Journal of*, 20(1):312 –325, feb. 2011.

[94] H Lee and K Chakrabarty. Test Challenges for 3D Integrated Circuits. *Design Test of Computers, IEEE*, PP(99):1, 2009.

[95] Xin Li, Yuchun Ma, and Xianlong Hong. A novel thermal optimization flow using incremental floorplanning for 3D ICs. In *ASPDAC*, pages 347–352. IEEE Press, 2009.

[96] S.K. Lim. Physical design for 3D system on package. *Design Test of Computers, IEEE*, 22(6):532 – 539, nov.-dec. 2005.

[97] Chang-Tzu Lin, De-Sheng Chen, and Yi-Wen Wang. GPE: a new representation for VLSI floorplan problem. In *Computer Design: VLSI in Computers and Processors, 2002. Proceedings. 2002 IEEE International Conference on*, pages 42 – 44, 2002.

[98] Jai-Ming Lin and Yao-Wen Chang. TCG: a transitive closure graph-based representation for non-slicing floorplans. In *Proceedings of the 38th annual Design Automation Conference*, DAC '01, pages 764–769, New York, NY, USA, 2001. ACM.

[99] C.C. Liu, I. Ganusov, M. Burtscher, and Sandip Tiwari. Bridging the processor-memory performance gap with 3D IC technology. *Design Test of Computers, IEEE*, 22(6):556 – 564, nov.-dec. 2005.

[100] Jing Liu, Weicai Zhong, Licheng Jiao, and Xue Li. Moving block sequence and organizational evolutionary algorithm for general floorplanning with arbitrarily shaped rectilinear blocks. *Evolutionary Computation, IEEE Transactions on*, 12(5):630 –646, oct. 2008.

[101] Gabriel H. Loh and Yuan Xie. 3D Stacked Microprocessor: Are We There Yet? *Micro, IEEE*, 30(3):60 –64, may-june 2010.

[102] Gabriel H. Loh, Yuan Xie, and Bryan Black. Processor Design in 3D Die-Stacking Technologies. *Micro, IEEE*, 27(3):31 –48, may-june 2007.

[103] Manuel Lozano, Francisco Herrera, and José Ramón Cano. Replacement strategies to preserve useful diversity in steady-state genetic algorithms. *Inf. Sci.*, 178(23):4421–4433, December 2008.

[104] Jian-Qiang Lu. 3-D Hyperintegration and Packaging Technologies for Micro-Nano Systems. *Proceedings of the IEEE*, 97(1):18 –30, jan. 2009.

[105] Thé Van Luong, Nouredine Melab, and El-Ghazali Talbi. GPU-based island model for evolutionary algorithms. In *Proceedings of the 12th annual conference on Genetic and evolutionary computation*, GECCO '10, pages 1089–1096, New York, NY, USA, 2010. ACM.

[106] Qiang Ma and Evangeline F. Y. Young. Voltage island-driven floorplanning. In *Proceedings of the 2007 IEEE/ACM international conference on Computer-aided design*, ICCAD '07, pages 644–649, Piscataway, NJ, USA, 2007. IEEE Press.

[107] K.F. Man, K.S. Tang, and S. Kwong. Genetic algorithms: concepts and applications [in engineering design]. *Industrial Electronics, IEEE Transactions on*, 43(5):519 –534, oct 1996.

[108] Danilo P. Mandic and Jonathon Chambers. *Recurrent Neural Networks for Prediction: Learning Algorithms,Architectures and Stability*. John Wiley & Sons, Inc., New York, NY, USA, 2001.

[109] Zbigniew Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs*. Springer, 1996.

[110] Sun Microsystems. http://www.oracle.com/technetwork/systems/opensparc/index.html, 2007.

[111] D. Millán-Ruiz and J.I. Hidalgo. Migration and replacement policies for preserving diversity in dynamic environments. In *Proceedings of the European Conference on the Applications of Evolutionary Computation*, GECCO '05, 2012.

[112] David Millán-Ruiz and José Ignacio Hidalgo. A memetic algorithm for workforce distribution in dynamic multi-skill call centres. In *EvoCOP*, pages 178–189, 2010.

[113] G. E. Moore. Cramming More Components onto Integrated Circuits. *Electronics*, 38(8):114–117, April 1965.

[114] Pablo Moscato. On evolution, search, optimization, genetic algorithms and martial arts - towards memetic algorithms, 1989.

[115] M. Motoyoshi. Through-Silicon Via (TSV). *Proceedings of the IEEE*, 97(1):43 –48, jan. 2009.

[116] Srinivasan Murali, Ciprian Seiculescu, Luca Benini, and Giovanni De Micheli. Synthesis of networks on chips for 3D systems on chips. In *Proceedings of the 2009 Asia and South Pacific Design Automation Conference*, ASP-DAC '09, pages 242–247, Piscataway, NJ, USA, 2009. IEEE Press.

[117] H. Murata, K. Fujiyoshi, S. Nakatake, and Y. Kajitani. Rectangle-packing-based module placement. In *Computer-Aided Design, 1995. ICCAD-95. Digest of Technical Papers., 1995 IEEE/ACM International Conference on*, pages 472 –479, nov 1995.

[118] R.K. Nain and M. Chrzanowska-Jeske. Placement-aware 3D floorplanning. In *Circuits and Systems, 2009. ISCAS 2009. IEEE International Symposium on*, pages 1727 –1730, may 2009.

[119] NVIDIA. http://www.nvidia.com, 2011.

[120] N. Okada, C. Kodama, T. Sato, and K. Fujiyoshi. Thermal Driven Module Placement Using Sequence-pair. In *Circuits and Systems, 2006. APCCAS 2006. IEEE Asia Pacific Conference on*, pages 1871 –1874, dec. 2006.

[121] OpenSPARC. http://www.opensparc.net/pubs/preszo/07/n2isscc.pdf, 2007.

[122] V. Pareto. *Cours D'Economie Politique*. F. Rouge, Lausanne, Switzerland, 1896.

[123] Petr Pospíchal. GPU-Based Acceleration of the Genetic Algorithm. In *Proceedings of the 16th Conference Student EEICT 2010 Volume 5*, pages 234–238. Faculty of Information Technology BUT, 2010.

[124] P. Ramm, A. Klumpp, J. Weber, N. Lietaer, M. Taklo, W. De Raedt, T. Fritzsch, and P. Couderc. 3D Integration technology: Status and application development. In *ESSCIRC, 2010 Proceedings of the*, pages 9 –16, sept. 2010.

[125] I. Rechenberg. *Evolutionsstrategie: optimierung technischer systeme nach prinzipien der biologischen evolution*. Frommann-Holzboog, 1973.

[126] José L. Risco-Martín, David Cuesta, and José L. Ayala. Niagara 3D heterogeneous benchmarks. Available at: http://bioinspired.dacya.ucm.es/lib/exe/fetch.php?media=n3hcnnn.zip, 2012.

[127] José L. Risco-Martín, J. Ignacio Hidalgo, Juan Lanchares, and Oscar Garnica. Solving discrete deceptive problems with EMMRS. In *Proceedings of the 10th annual conference on Genetic and evolutionary computation*, GECCO '08, pages 1139–1140, New York, NY, USA, 2008. ACM.

[128] M. Ruciński, D. Izzo, and F. Biscani. On the impact of the migration topology on the island model. *Parallel Comput.*, 36(10-11):555–571, October 2010.

[129] Stephen Rusu. Presentation of the International Solid-State Circuits Conference 2013 (ISSCC). http://www.electroiq.com/articles/sst/2013/02/isscc-2013–high-performance-digital-trends.html, 2013.

[130] Mohamed M. Sabry, David Atienza, and Ayse K. Coskun. Thermal analysis and active cooling management for 3D MPSoCs. In *Circuits and Systems (ISCAS), 2011 IEEE International Symposium on*, pages 2237 –2240, may 2011.

[131] Karthik Sankaranarayanan, Sivakumar Velusamy, Mircea Stan, Charles L, and Kevin Skadron. A case for thermal-aware floorplanning at the microarchitectural level. *JILP*, 7(1):8–16, 2005.

[132] J. David Schaffer. Multiple Objective Optimization with Vector Evaluated Genetic Algorithms. In *Proceedings of the 1st International Conference on Genetic Algorithms*, pages 93–100, Hillsdale, NJ, USA, 1985. L. Erlbaum Associates Inc.

[133] H. P. Schwefel. *Numerische Optimierung von Computer-Modellen mittels Evolutionsstrategie*. Birkhäuser Verlag, Basel, 1974.

[134] Robert Sedgewick and Kevin Wayne. *Algorithms in Java*. Addison-Wesley Professional, 2011.

[135] P.H. Shiu, R. Ravichandran, S. Easwar, and S.K. Lim. Multi-layer floorplanning for reliable system-on-package. In *Circuits and Systems, 2004. ISCAS '04. Proceedings of the 2004 International Symposium on*, volume 5, pages V–69 – V–72 Vol.5, may 2004.

[136] H.K. Singh, A. Isaacs, and T. Ray. A pareto corner search evolutionary algorithm and dimensionality reduction in many-objective optimization problems. *Evolutionary Computation, IEEE Transactions on*, 15(4):539 –556, aug. 2011.

[137] Hiroshi Someya and Masayuki Yamamura. A genetic algorithm without parameters tuning and its application on the floorplan design problem. In *Proceedings of Genetic and Evolutionary Computation Conference: GECCO 99*, pages 620–627, 1999.

[138] A. Sridhar, A. Vincenzi, M. Ruggiero, T. Brunschwiler, and D. Atienza. 3D-ICE: Fast compact transient thermal modeling for 3D ICs with inter-tier liquid cooling. In *Computer-Aided Design (ICCAD), 2010 IEEE/ACM International Conference on*, pages 463 –470, nov. 2010.

[139] Arvind Sridhar, Alessandro Vincenzi, Martino Ruggiero, and David Atienza Alonso. Neural Network-Based Thermal Simulation of Integrated Circuits on GPUs. *IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems*, 31(1):23–36, 2012.

[140] N. Srinivas and Kalyanmoy Deb. Multiobjective Optimization Using Nondominated Sorting in Genetic Algorithms. *Evolutionary Computation*, 2:221–248, 1994.

[141] T. Takahashi, P.N. Guo, C.K. Cheng, and T. Yoshimura. Floorplanning using a tree representation: a summary. *Circuits and Systems Magazine, IEEE*, 3(2):26 – 29, 2003.

[142] Maolin Tang and Alvin Sebastian. A Genetic Algorithm for VLSI Floorplanning Using O-Tree Representation. In Franz Rothlauf, Jurgen Branke, Stefano Cagnoni, David Corne, Rolf Drechsler, Yaochu Jin, Penousal Machado,

Elena Marchiori, Juan Romero, George Smith, and Giovanni Squillero, editors, *Applications of Evolutionary Computing*, volume 3449 of *Lecture Notes in Computer Science*, pages 215–224. Springer Berlin / Heidelberg, 2005.

[143] Maolin Tang and Xin Yao. A Memetic Algorithm for VLSI Floorplanning. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 37(1):62 –69, feb. 2007.

[144] A.W. Topol, D.C. La Tulipe, L. Shi, S.M. Alam, D.J. Frank, S.E. Steen, J. Vichiconti, D. Posillico, M. Cobb, S. Medd, J. Patel, S. Goma, D. DiMilia, M.T. Robson, E. Duch, M. Farinelli, C. Wang, R.A. Conti, D.M. Canaperi, L. Deligianni, A. Kumar, K.T. Kwietniak, C. D'Emic, J. Ott, A.M. Young, K.W. Guarini, and M. Ieong. Enabling SOI-based assembly technology for three-dimensional (3D) integrated circuits (ICs). In *Electron Devices Meeting, 2005. IEDM Technical Digest. IEEE International*, pages 352 –355, dec. 2005.

[145] C. Torregiani, H. Oprins, B. Vandevelde, E. Beyne, and I. De Wolf. Compact thermal modeling of hot spots in advanced 3D-stacked ICs. In *Electronics Packaging Technology Conference, 2009. EPTC 09. 11th*, pages 131 –136, dec. 2009.

[146] Ching-Han Tsai and Sung-Mo Kang. Cell-level placement for improving substrate thermal distribution. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 19(2):253 –266, feb 2000.

[147] Shigeyoshi Tsutsui and Noriyuki Fujimoto. Solving quadratic assignment problems by genetic algorithms with GPU computation: a case study. In *Proceedings of the 11th Annual Conference Companion on Genetic and Evolutionary Computation Conference: Late Breaking Papers*, GECCO '09, pages 2523–2530, New York, NY, USA, 2009. ACM.

[148] R.R. Tummala. SOP: what is it and why? A new microsystem-integration technology paradigm-Moore's law for system integration of miniaturized convergent systems of the next decade. *Advanced Packaging, IEEE Transactions on*, 27(2):241 – 249, may 2004.

[149] G. Van der Plas, P. Limaye, I. Loi, A. Mercha, H. Oprins, C. Torregiani, S. Thijs, D. Linten, M. Stucchi, G. Katti, D. Velenis, V. Cherman, B. Vandevelde, V. Simons, I. De Wolf, R. Labie, D. Perry, S. Bronckers, N. Minas, M. Cupac, W. Ruythooren, J. Van Olmen, A. Phommahaxay, M. de Potter de ten Broeck, A. Opdebeeck, M. Rakowski, B. De Wachter, M. Dehan, M. Nelis, R. Agarwal, A. Pullini, F. Angiolini, L. Benini, W. Dehaene, Y. Travaly, E. Beyne, and P. Marchal. Design Issues and Considerations for Low-Cost 3-D TSV IC Technology. *Solid-State Circuits, IEEE Journal of*, 46(1):293 –307, jan. 2011.

[150] V. Černý. Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm. *Journal of Optimization Theory and Applications*, 45(1):41–51, January 1985.

[151] Sebastien Verel, Arnaud Liefooghe, and Clarisse Dhaenens. Set-based multiobjective fitness landscapes: a preliminary study. In *Proceedings of the 13th annual conference on Genetic and evolutionary computation*, GECCO '11, pages 769–776, New York, NY, USA, 2011. ACM.

[152] Guan Wang, Degang Wu, and K.Y. Szeto. Quasi-parallel genetic algorithms with different communication topologies. In *Evolutionary Computation (CEC), 2011 IEEE Congress on*, pages 721 –727, june 2011.

[153] Jean-Paul Watson. An Introduction to Fitness Landscape Analysis and Cost Models for Local Search. In Michel Gendreau and Jean-Yves Potvin, editors, *Handbook of Metaheuristics*, volume 146 of *International Series in Operations Research and Management Science*, pages 599–623. Springer US, 2010.

[154] Edward D. Weinberger. NP Completeness of Kauffman's N-k Model, A Tuneable Rugged Fitness Landscape. Working papers, Santa Fe Institute, 1996.

[155] D.F. Wong and C.L. Liu. A new algorithm for floorplan design. In *Design Automation, 1986. 23rd Conference on*, pages 101 – 107, june 1986.

[156] Man Leung Wong. Parallel multi-objective evolutionary algorithms on graphics processing units. In *Proceedings of the 11th Annual Conference Companion*

on Genetic and Evolutionary Computation Conference: Late Breaking Papers, GECCO '09, pages 2515–2522, New York, NY, USA, 2009. ACM.

[157] Bin Wu and Peng Li. Load-aware stochastic feedback control for DVFS with tight performance guarantee. In *VLSI and System-on-Chip (VLSI-SoC), 2012 IEEE/IFIP 20th International Conference on*, pages 231 –236, oct. 2012.

[158] Y. Yano and M. Kaneko. Solution space reduction of sequence pairs using model placement. In *Circuits and Systems, 2007. MWSCAS 2007. 50th Midwest Symposium on*, pages 1130 –1133, aug. 2007.

[159] Albert M. Young and Steven J. Koester. *Three-Dimensional Integrated Circuit Design*, chapter 3D Process Technology Considerations, pages 15–32. Springer Publishing Company, Incorporated, 2009.

[160] Xiuyi Zhou, Jun Yang, Yi Xu, Youtao Zhang, and Jianhua Zhao. Thermal-Aware Task Scheduling for 3D Multicore Processors. *Parallel and Distributed Systems, IEEE Transactions on*, 21(1):60 –71, jan. 2010.

[161] E. Zitzler, M. Laumanns, and L. Thiele. SPEA2: Improving the strength pareto evolutionary algorithm for multiobjective optimization. In K. C. Giannakoglou, D. T. Tsahalis, J. Périaux, K. D. Papailiou, and T. Fogarty, editors, *Evolutionary Methods for Design Optimization and Control with Applications to Industrial Problems*, pages 95–100, Athens, Greece, 2001. International Center for Numerical Methods in Engineering.

[162] E. Zitzler and L. Thiele. Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach. *IEEE Transactions on Evolutionary Computation*, 3(4):257–271, 1999.

Appendix A

Resumen

A.1 Introducción

En 2006, Dr. Wang, el entonces director ejecutivo de Samsung Electronics dijo: "La rápida adopción de la tecnología de integración 3D parece esencial e inevitable" [71]. Desde entonces se han realizado importantes avances en este paradigma tecnológico y se han comercializado distintos productos. Sin embargo, todavía no se ha alcanzado todo el potencial de la integración en tres dimensiones.

Aunque la integración en 3D se propuso a principios de la década de 1980, la propuesta pasó desapercibida hasta los años 2000 y siguientes, cuando empezaba a quedar claro el límite físico de los dispositivos fabricados en silicio. Hoy en día se está invirtiendo grandes cantidades de recursos en mejorar las tecnologías necesarias para la fabricación de arquitecturas tridimensionales. Para explicar esta tendencia, es necesario recapitular la evolución y las decisiones estratégicas de la industria de semiconductores desde mediados de los noventa. Desde entonces, no sólo el número de chips fabricados se ha disparado sino que el gran poder computacional alcanzado se encuentra al alcance de la mano con el desarrollo masivo de dispositivos móviles. No obstante, el camino no ha sido fácil para ingenieros y diseñadores, ya que éstos han tenido que lidiar con numerosas dificultades. Como se detalla a continuación, algunos de los problemas encontrados se han resuelto de manera brillante mientras que otros tal vez hayan sido pospuestos a un futuro cercano.

Inicialmente, las innovaciones arquitectónicas junto con el escalado de frecuencia permitieron grandes incrementos en el rendimiento. Sin embargo el aumento de la frecuencia de reloj se fue frenando a principio de la década del 2000 debido a que, sobrepasado un cierto umbral, el aumento de frecuencia deja de ser eficiente desde un punto de vista energético. Además, las altas temperaturas alcanzadas son perjudiciales para la fiabilidad y la vida media de los dispositivos de silicio.

Por lo tanto, la industria se decantó hacia arquitecturas dotadas de múltiples procesadores con una menor frecuencia de funcionamiento. Así, pese a que la frecuencia se estancase o incluso disminuyera, el número de instrucciones ejecutadas por segundo continuó creciendo.

La siempre creciente necesidad de mayor rendimiento llevó a incrementar el número de procesadores por chip. Este aumento fue posible gracias a la reducción progresiva del tamaño de los transistores, lo que permitía una mayor densidad de integración. Así, chips de generaciones consecutivas presentaban una área similar. Típicamente se ha doblado el número de procesadores en cada generación debido a la estrategia de replicación adoptada, resultando en una mayor demanda de ancho de banda. Por otra parte, procesadores recientes han reincorporado técnicas de Multithreading Simultáneo, lo que conlleva una mayor virtualización y una todavía mayor necesidad de ancho de banda ya que se realizan más transacciones de bus.

Al mismo tiempo, las mejoras realizadas para reducir los retardos debidos al cableado no han seguido el ritmo marcado por el incremento de frecuencia. Este problema es cada vez más relevante dado que se espera un incremento del área de los chips debido al alcance del límite físico de la densidad de integración. En efecto, a medida que se reduce el tamaño de los transistores, la corriente de fuga aumenta y la variabilidad empeora. Ya que no será posible seguir reduciendo el tamaño de los transistores, el área requerida para integrar un mayor número de procesadores aumentará siginificativamente. Como consecuencia, el porcentaje de área del chip accesible en un ciclo seguirá disminuyendo en cada generación. Esta reducción, a su vez, conlleva mayores latencias por transacciones de bus. Así, el mayor número de transacciones y su mayor latencia limitarán el incremento de rendimiento propor-

cionado por la integración de un mayor número de procesadores en un sólo chip [16].

En resumen, se puede decir que ya no será posible obtener incrementos de rendimiento de forma sencilla. La industria de los semiconductores tendrá por lo tanto que realizar cambios drásticos para responder a la demanda de redimiento [91]. Integrar un mayor número de procesadores en un sólo chip sólo será beneficioso si se proporciona un mayor ancho de banda y se reduce la distancia entre componentes conectados. Aquí es donde la integración 3D entra en escena. En efecto, apilar circuitos integrados conectados verticalmente permite reducir el cableado global y proporciona un mayor ancho de banda entre las capas conectadas. Por ejemplo, un diseño particionado en dos capas de un Pentium IV resultó en un aumento del rendimiento de un 15% gracias a un mejor diseño de los pipelines [102]. A grandes rasgos, en arquitecturas 3D, se espera que el cableado total disminuya en un factor de $(N_{capas})^{1/2}$, mientras que en las arquitecturas 2D éste aumenta cuadráticamente [44]. Además, las conexiones verticales entre capas implementadas mediante Through Silicon Vias (TSVs) proporcionan un acho de banda masivo. Esta mayor capacidad puede explotarse, entre otras aplicaciones, para acelerar los accesos a memoria si procesadores y memorias están conectados mediante TSVs.

Por lo tanto, la integración 3D se considera una tecnología clave para hacer frente a la demanda de rendimiento en los próximos años. Dicha tecnología en combinación con la reducción del tamaño de los transistores se consideró inicialmente como una vía para alcanzar niveles de integración más altos que los predecidos por la Ley de Moore [113] ("More than Moore" [101]). Además, el diseño en tres dimensiones puede llevar a otras mejoras. En efecto, la reducción del cableado también acarrea un descenso del consumo de potencia y elimina problemas de timing debidos a fenómenos de clock skew y de jitter [102]. Por otra parte, se posibilita la integración heterogénea ya que diferentes procesos tecnológicos pueden ser combinados en una misma plataforma. Finalmente, la integración modular de las distintas capas puede reducir el coste global y el esfuerzo de diseño [52]. En resumen, la integración 3D proporciona:

1. Mayor rendimiento debido a la reducción de la longitud del cableado y al mayor ancho de banda proporcionado por las TSVs.

2. Disminución del consumo de potencia gracias a una reducción de la longitud de las redes de reloj y de alimentación (comparando con una arquitectura equivalente 2D.

3. Integración heterogénea.

4. Integración modular, reduciendo el esfuerzo de diseño.

Como consecuencia, la integración en 3D ha suscitado un gran interés tanto en el mundo de la industria como en el académico. La Tabla A.1 muestra algunas de las organizaciones involucradas en proyectos de investigación y desarrollo realcionados con la integración en 3D (ver [124]).

AMERICA	**ASIA**	**EUROPE**
Albany Nanotech	Amkor	3D-PLUS
Cubic Wafer	ASE	CEA-LETI
Freescale	ASET	EMFT Munich
Ga Tech	Chartered	Fraunhofer IZM
IBM	Elpida	IMEC
Intel	Fujikura	Infineon
Lincoln Labs	Hitachi	NXP
Micro	IME	Sensonor
MIT	ITRI	Siemens
NC State	KAIST	SINTEF
RPI	NEC	STM
RTI	Oki	TU Chemnitz
SANDIA National Labs	Renesas	VTI
Sematech	Samsung	
Stanford	Sanyo	
Texas Instruments	Sharp	
Tezzaron	Sony	
Univ. Arkansas	STATSChipPAC	
Univ. Minn	Tohoku Univ.	
Xilinx	Toshiba	
Ziptronix	TSMC	
	ZyCube	

Table A.1: Actividad relacionada con la integración en 3D según lo expuesto en [124]

La integración 3D engloba varias alternativas tecnológicas que dan lugar a una gran variedad de productos finales. Algunas aplicaciones comerciales basadas en un empaquetado simple ya están comercializadas. Sin embargo, todavía no ha comenzado la producción en masa de Circuitos Integrados 3D debido a que los requisitos tecnológicos no pueden ser satisfechos con costes lo suficientemente reducidos [124]. Por otra parte, el salto necesario para adoptar la tecnología 3D constituye una difícil decisión estratégica por parte de la industria ya que los riesgos asociados son mayores que en cambios tecnológicos anteriores. Varias de las etapas del proceso de fabricación tienen que ser mejoradas, tales como la formación de vias, el alineamiento de los wafers, o su unión [92]. Otras dificultades están relacionadas con los procedimientos de test [94] y la falta de herramientas de Electronic Design Automation (EDA) adecuadas para el diseño 3D [104], [92]. En efecto, la industria necesita la creación de estandars y de una suite de herramientas para facilitar la integración 3D. En particular, se necesitan urgentemente algoritmos de place-and-route en 3D, modelos de timing y de potencia y métodos de floorplanning [56]. La Figura A.1 muestra el estado de la Investigación y Desarrollo de la integración 3D en el año 2009 [94].

Como se muestra en la figura, existe una carencia de herramientas de diseño automático. Estas herramientas deben evitar temperaturas altas y elevados gradientes térmicos ya que estos dos factores perjudican el rendimiento, la fiabilidad y la vida media de los chips [44], [90]. Estos problemas críticos se magnifican en entornos 3D por dos motivos principales. En primer lugar, la densidad de potencia aumenta con el número de capas, ya que se obtiene una mayor densidad de integración. En segundo lugar, las capas adicionales se apilan cada vez más lejos de la base del chip, siendo más difícil la disipación de calor. Por ello, lidiar con las restricciones térmicas es de vital importancia para la viabilidad de la fabricación a gran escala de productos basados en integración 3D [102], [52], [104].

El floorplanning térmico se emplea para reducir la temperatura máxima del chip, tratando de encontrar una disposición óptima de los componentes que minimice área, temperatura y cableado (traducido en rendimiento). La idea principal que motiva esta técnica es que disposiciones estrictamente regulares no presentan las

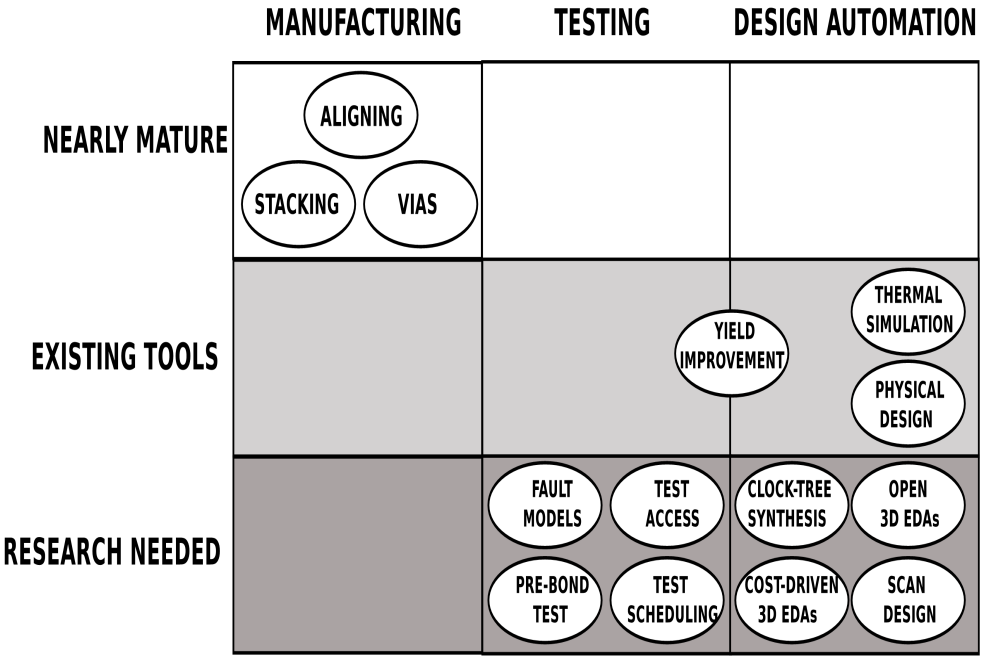


Figure A.1: Estado de la Investigación y Desarrollo de la integración 3D

características térmicas requeridas. En efecto, si un bloque caliente se ubica entre bloques fríos, tiene lugar una difusión lateral del calor y, como resultado, se reduce la temperatura del bloque caliente [131].

Las técnicas de floorplanning térmico son cada vez más necesarias ya que la reducción del tamaño de los transistores y el escalado de frecuencia han llevado a alcanzar altas densidades de potencia que provocan temperaturas extremas. Existe una gran variedad de propuestas que atacan la optimización de circuitos lógicos que han sido extendidas para lidiar con plataformas 3D. Sin embargo, estos métodos no suelen proporcionar en un tiempo reducido configuraciones óptimas desde un punto de vista térmico. Como se demostrará en este trabajo, existe una necesidad de nuevos métodos idóneos para el nuevo entorno tridimensional.

Aunque todavía no se ha introducido una suite completa de herramientas que cubran todas las etapas del diseño de chips 3D, ya existen algunas propuestas útiles para realizar tareas de exploración arquitectónica. Por ejemplo, se han propuesto distintos simuladores térmicos, generalmente empleados para garantizar que las con-

figuraciones estudiadas no alcanzan temperaturas extremas ni elevados gradientes térmicos [75], [24], [145], [149] y [138]. Otro ejemplo es la simulación de refrigeración líquida necesaria para un gran número de arquitecturas 3D, en particular para los MPSoCs en 3D [9], [41] y [130]. Como quedará de manifiesto en este trabajo, el enfriamiento proporcionado por esta técnica de refrigeración ha de ser considerado en el proceso de optimización para obtener configuraciones que presenten simultáneamente un rendimiento y una temperatura óptimos.

Desde el punto de vista de la complejidad computacional, el problema del floorplanning se clasifica com NP-Duro (ver [17] y [58]). Añadir una tercera dimensión resulta en un espacio de diseño todavía mayor ya que se explora un una mayor variedad de configuraciones. Además, las plataformas estudiadas presentan un compromiso entre perfil térmico y rendimiento, ambas características fundamentales de las plataformas 3D. Por lo tanto, se necesitan métodos apropiados para explorar el espacio de soluciones de manera eficiente y así obtener soluciones optimizadas tanto en temperatura como en rendimiento.

En este trabajo, se utilizan Algoritmos Evolutivos Multiobjetivo (AEMOs) para atacar el problema del floorplanning térmico de MPSoCs 3D. Estos algoritmos constituyen una extensión de los Algoritmos Evolutivos (AEs). En breve, los AEs son metaheurísticas poblacionales inspiradas el el concepto Darwinista de evolución capaces de obtener buenas soluciones de problemas complejos en un corto periodo de tiempo. Por su parte, los AEMOs incorporan el concepto de optimalidad de Pareto y constituyen una herramienta eficiente para encontrar soluciones con un buen compromiso entre dos o más objetivos conflictivos. Por lo tanto, los AEMOs son una herramienta idónea para optimizar las arquitecturas estudiadas, obteniendo configuraciones que minimizan simultáneamente temperatura y cableado.

Dada la importancia de los MPSoCs 3D y la falta de herramientas para su optimización, establecemos los siguientes objetivos:

- Confirmar la viabilidad de estas arquitecturas, verificando para que se mejora el rendimiento y que las temperaturas máximas se mantienen en niveles aceptables.

- Estudiar la idoneidad de propuestas de floorplanning existentes en la literatura adpatadas para la optimización de las arquitecturas estudiadas. De la misma manera, se han de comparar las técnicas ya validades con nuevas representaciones y algoritmos.

- Es necesario proporcionar una gran diversidad de soluciones óptimas con un buen compromiso entre rendimiento y temperatura. Así, los fabricantes de estas arquitecturas podrán elegir la mejor configuración de acuerdo con sus propios criterios.

- Proponer estrategias de floorplanning paralelas para acelerar el proceso de exploración arquitectónica.

- Comparar la idoneidad de varios modelos térmicos existentes en un contexto de floorplanning.

- Estudiar la necesidad de otras técnicas de enfriamiento como la refrigeración líquida. En caso de ser necesario, considerar el efecto de dicha refrigeración en el proceso de optimización.

Una vez motivado el estudio del floorplanning térmico de MPSoCs 3D y habiendo fijado los objetivos del trabajo, pasamos a describir la estructura de esta tesis con una breve introducción de los capítulos que la componen.

En el Capítulo 2, motivamos el estudio de MultiProcessor Systems-on-Chip en tres dimensiones. Para ello, presentamos las tecnologías existentes y estudiamos su compatibilidad con diferentes enfoques de diseño. También se introducen en este capítulo varios modelos térmicos empleados a lo largo del trabajo para obtener la reacción térmica de las arquitecturas estudiadas.

A continuación, en el Capítulo 3, presentamos las propuestas de floorplanning más relevantes adaptadas para optimizar tres arquitecturas complejas basadas en la plataforma Niagara. El estudio comparativo evidencia la necesidad de nuevas técnicas para lidiar con las restricciones térmicas impuestas por los multiproce-

sadores de tres dimensiones.

Los Algoritmos Evolutivos Multiobjetivo utilizados en este trabajo para atacar el problema del floorplanning térmico se explican el el Capítulo 4. Con el fin de introducir estos métodos, se proporciona una introducción a la optimización multiobjetivo. También se introducen los Algoritmos Evolutivos y las principales técnicas de paralelización empleadas para acelerarlos.

En el Capítulo 5 se recapitulan varias contribuciones de esta tesis. En primer lugar se presenta una versión paralela de un floorplanner evolutivo multiobjetivo con el fin de acelerar el proceso de optimización. También se introduce conocimiento al problema por medio de perfiles de potencia obtenidos mediante simulaciones en una etapa previa para guiar el proceso de optimización. Por último, se compara la optimización térmica y el rendimiento alcanzados utilizando diferentes modelos térmicos en el proceso de floorplanning.

La nueva representación Direct Mapping se presenta en el Capítulo 6. El uso de dicha representación en conjunción con un AEMO diseñado para realizar la optimización de MPSoCs 3D lleva a mejorar las propuestas anteriores de floorplanning. Además, el algoritmo propuesto está guiado por simulaciones térmicas que incluyen el efecto refrigerador de microcanales ubicados en capas intermedias del chip.

Aunque se han incluido conclusiones parciales en los capítulos previos, las más relevantes están resumidas en el Capítulo 7. También se comentan posibles mejoras y trabajos futuros. Por último, se proporcionan las referencias a las publicaciones asociadas a esta tesis, así como a los proyectos y becas con los que este trabajo ha sido subvencionado.

A.2 Principales Contribuciones

En este trabajo, atacamos el problema del floorplanning térmico con área fija. El problema estudiado es por lo tanto equivalente a un problema de ubicación de bloques en el que se deben considerar restricciones térmicas y de rendimiento. El

problema de la colocación de bloques se formula como sigue:

Problema de la ubicación de bloques

Todos los bloques que modelan los distintos componentes del sistema deben ser colocados en la pila 3D, que impone una longitud máxima L, un ancho W y una altura H. Cada bloque B_i $(i = 1, 2, \ldots, n)$ del modelo se caracteriza por un ancho w_i, una altura h_i y una longitud l_i. Es importante notar que en los casos estudiados en este trabajo, $h_i = 1$ para todos los componentes. Definimos el vector (x_i, y_i, z_i) como las coordenadas geométricas del bloque B_i, donde $0 \leq x_i \leq L - l_i$, $0 \leq y_i \leq W - w_i$, $0 \leq z_i < H$ (véase la Figura A.2). Se emplea (x_i, y_i, z_i) para denotar la esquina izquierda inferior del bloque B_i mientras que se supone que las coordenadas de la esquina inferior izquierda del chip son $(0, 0, 0)$. A diferencia de los enfoques tradicionales de problemas de floorplanning en 2D, no se trata de minimizar el área ya que ésta se fija de antemano.

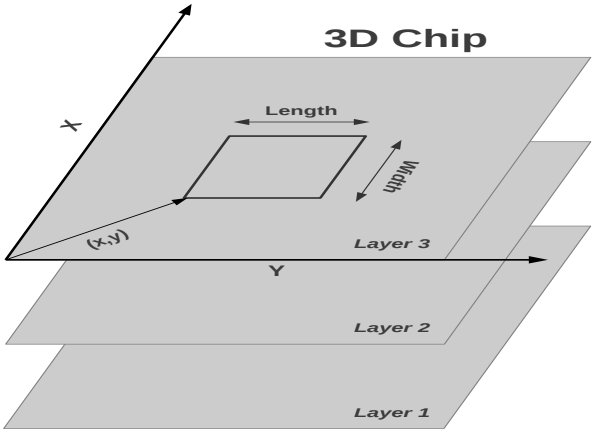


Figure A.2: Block representation

En esta sección se exponen brevemente las diferentes contribuciones de esta tesis:

- Se proporciona la descripción del algoritmo llamado Multi-objective Floorplanning Algorithm (MFA), presentado por primera vez en [38], ya que éste representa el punto de partida de este trabajo.
- Se presenta una implementación multi-hilo siguiendo un modelo Maestro-Trabajador para reducir el tiempo de ejecución del proceso de floorplanning

- Se presenta una estrategia que permite introducir conocimiento en el proceso de floorplanning. Así, se obtienen perfiles dinámicos de consumo de potencia en una etapa previa al proceso de floorplanning con el fin de guiar mejor la optimización térmica.

- Comparamos el rendimiento y la optimización térmica logrados al introducir diferentes modelos térmicos en el proceso de optimización.

- Proponemos la representación Direct Mapping junto con un AEMO diseñado para realizar la optimización de MPSoCs 3D.

- Estudiamos el impacto en las configuraiones finales de considerar técnicas de refrigeración líquida durante el proceso de optimización.

A.2.1 Multi-Objective Floorplanning Algorithm

Multi-Objective Floorplanning Algorithm (MFA) es un algoritmo evolutivo multiobjetivo basado en NSGA-II (véase el Capítulo 4). Este floorplanner gestiona soluciones codificadas que mejoran gradualmente a medida que se desarrolla el proceso evolutivo, con el fin de proporcionar configuraciones óptimas desde un punto de vista del rendimiento y del perfil térmico. MFA se puede clasificar como un floorplanner híbrido ya que la heurística de decodificación implementa un floorplanning incremental inspirada en técnicas constructivas, mientras que el AEMO en que engloba dicha heurística es esencialmente iterativo. La Figura A.3 ilustra esta idea. Los detalles del algoritmo se proporcionan a continuación.

Para ilustrar el funcionamiento de MFA, analizamos la optimización de una plataforma de 48 núcleos heterogénea inspirada en la arquitectura Niagara. La configuración original de dicha plataforma se muestra en la Figura A.4. Se puede apreciar una disposición regular de los componentes de la arquitectura. Como consecuencia, los núcleos SPARC (SPC) son ubicados los unos encima de los otros produciendo puntos calientes. Por otro lado, la Figura A.5 muestra los mapas térmicos de las diferentes capas de una solución no dominada encontrada por MFA. Esta figura muestra una colocación optimizada de los núcleos SPARC (SPC), POWER6 (P6), memorias (L2) y Crossbars (Cross). En esta configuración los componentes

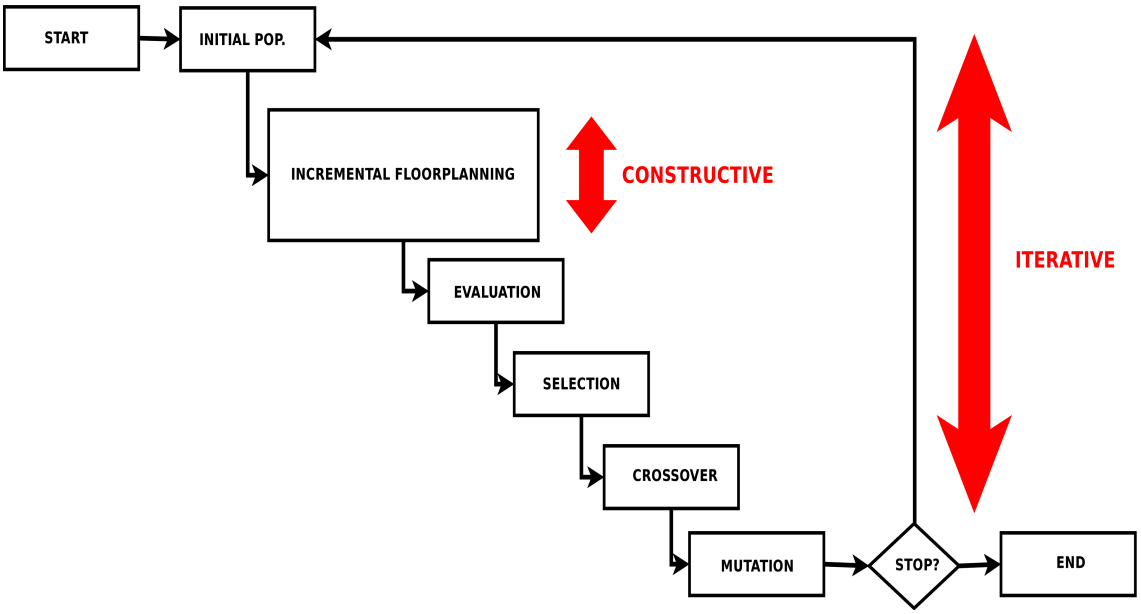


Figure A.3: MFA: un floorplanner híbrido

más calientes (los núcleos SPARC) han sido ubicados en los bordes del chip y en las capas exteriores, y tienden a estar separados entre sí. De hecho, el floorplanner evita la superposición vertical de éstos procesadores ya que la propagación vertical de calor vertical también se tiene en cuenta. Los crossbars se colocan en capas intermedias para minimizar la longitud global del cableado.

En la Sección 5.1 de esta tesis, se comprueba que las arquitecturas multiprocesador 3D actuales y del futuro a corto plazo requieren técnicas de floorplanning térmico capaces de reducir las temperaturas máximas alcanzadas. Se demuestra también que MFA proporciona configuraciones optimizadas para sistemas heterogéneos dotados de 48 y 128 núcleos. Además, se realiza un estudio de la convergencia de dicho floorplanner que revela un comportamiento adecuado del algoritmo en el proceso de optimización. Sin embargo, las técnicas actuales floorplanning tales como MFA que tienen en cuenta restricciones térmicas presentan un cuello de botella en la decodificación y evaluación de las soluciones. Por tanto, es necesario acelerar el proceso de floorplanning para permitir una exploración del espacio de soluciones más eficiente.

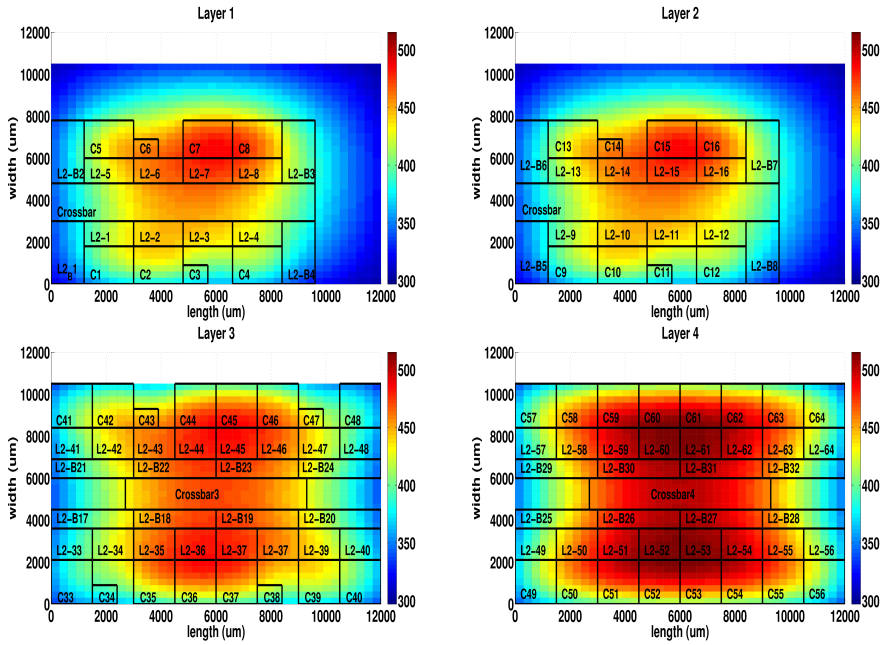


Figure A.4: Mapas térmicos de las 4 capas de la configuración original de la plataforma de 48 cores

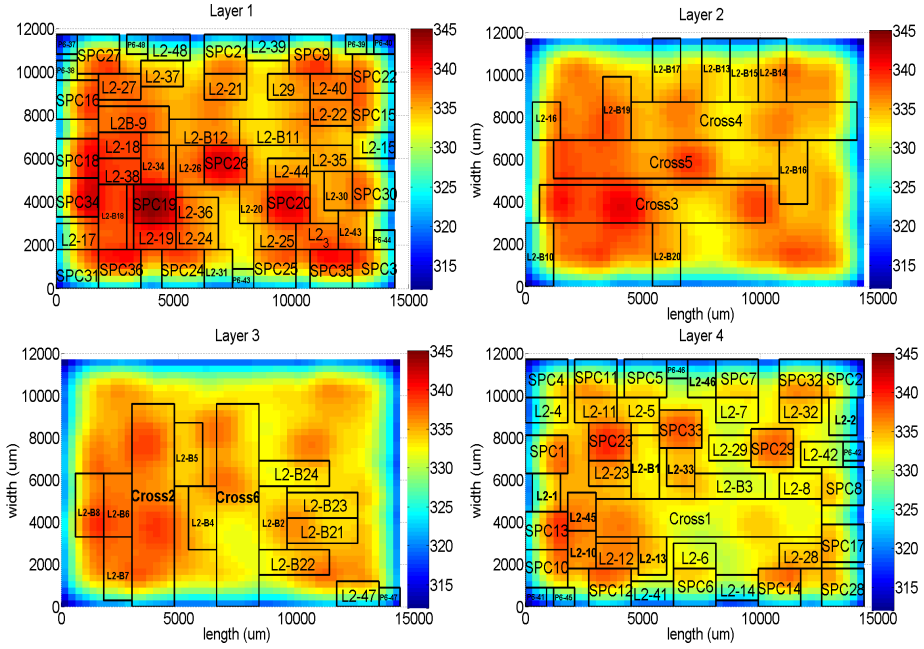


Figure A.5: Mapas térmicos de las 4 capas de la configuración optimizada de la plataforma de 48 cores

A.2.2 Floorplanner basado en el modelo Maestro-Trabajador

En la Sección 5.2, se presenta una implementación paralela de MFA siguiendo un modelo Maestro-Trabajador. El esfuerzo de paralelización está justificado ya que la

fase de evaluación del algoritmo conlleva más del 99% del tiempo de ejecución. Este cuello de botella aparece porque cada individuo de la población tiene que ser decodificado y evaluado térmicamente en todas las generaciones del proceso evolutivo.

Se emplea el modelo Maestro-Trabajador porque, aunque el fitness se establece mediante modelo térmico simplificado, el coste computacional de esta evaluación aumenta cuadráticamente con el número de componentes. Por lo tanto, es interesante para aprovechar el hecho de que los algoritmos evolutivos son intrínsecamente paralelos y llevar a cabo la evaluación de la población de manera concurrente. La Figura A.6 representa el enfoque utilizado en esta sección.

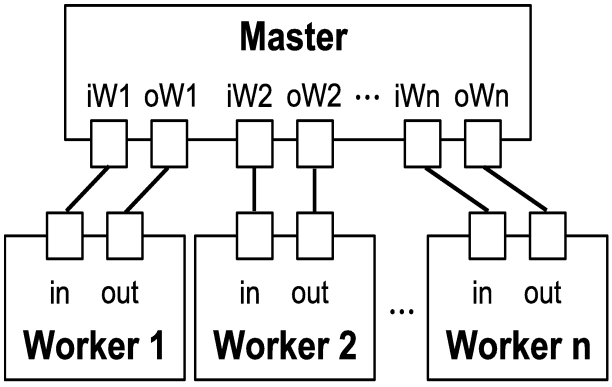


Figure A.6: Master-Worker configuration

El Maestro distribuye la población entre los n trabajadores y no lleva a cabo ninguna evaluación. Así, la carga computacional se divide en n tareas. Una vez los trabajadores han terminado su tarea, éstos envian el correspondiente resultado junto con el subconjunto de la población recibido al Maestro. Aunque el algoritmo se detiene y espera a que todos los trabajadores hayan terminado, esta implementación es claramente más rápida que la ejecución secuencial (siempre que cada subconjunto sea lo suficientemente grande como para compensar tiempos de comunicación).

Proponemos una implemetación multi-hilo en el que únicamente el maestro ejecuta el hilo principal del algoritmo. Dado que sólo los trabajadores ejecutan la evaluación de los diferentes subgrupos de la población, se obtiene una aceleración del orden del número de núcleos del procesador en el que se ejecuta el algoritmo.

Así, en nuestros experimentos ejecutados en un procesador de 4 núcleos, se han obtenido valores máximos de aceleración de 3.79 y 3.19 para las plataformas de 48 y 128 núcleos respectivamente con 5 trabajadores.

A.2.3 Floorplanner guiado por perfiles de potencia

En la Sección 5.3, se añade conocimiento al problema del floorplanning térmico por medio de una etapa de simulación en la que se obtienen perfiles de consumo potencia. De hecho, la temperatura de un chip dado depende de factores físicos tales como la disipación de potencia de los procesadores, el tamaño de las memorias, etc, pero también del perfil dinámico de las aplicaciones. Una de nuestras contribuciones es considerar perfiles de energía obtenidos mediante la simulación de aplicaciones reales ejecutadas en MPSoCs de grandes dimensiones. Es importante resaltar que tradicionalmente, este problema se ha enfocado considerando sólo el caso peor en términos de disipación de energía. Los perfiles de potencia se obtienen con OVPsim [76], un simulador de alto nivel para la exploración arquitectónica de multiprocesadores. En efecto, uno de los principales objetivos de este trabajo es comprender el efecto de la sincronización y la comunicación entre distintos procesadores en la temperatura de chips heterogéneos. La Figura A.7 muestra un patrón típico disipación de potencia de tres procesadores con diferente capacidad de cómputo trabajando juntos. Podemos ver claramente cómo la actividad del núcleo SPARC cambia periódicamente con el tiempo, ya que éste tiene que esperar a procesadores más lentos.

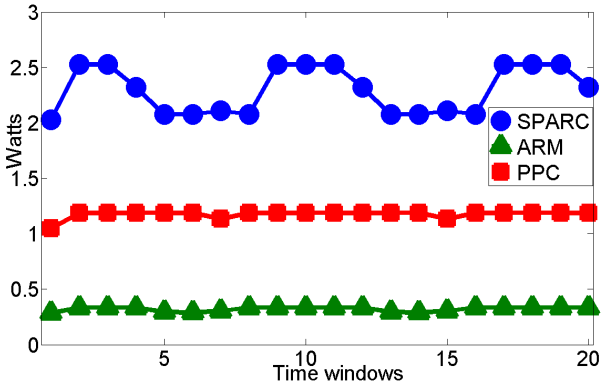


Figure A.7: Common power consumption pattern caused by synchronization

Los perfiles de potencia obtenidos mediante simulación se componen de 600 valores diferentes (100 ventanas temporales $\times$ 6 programas simulados). Obviamente, 600 objetivos son demasiados para un AEMO, ya que convergería demasiado despacio o no convergería en absoluto. Por lo tanto, se consideran cuatro estrategias para reducir el elevado número de objetivos. Para cada uno escenarios de optimización considerados, se obtienen cuatro configuraciones diferentes:

1) Como no tenemos ninguna configuración original para comparar las arquitecturas estudiadas, se propone como referencia una configuración optimizada únicamente en rendimiento, denominada BAS.

Para obtener las otras tres configuraciones, consideramos diferentes métricas térmicas computadas a partir de los datos recuperados de la simulación de 100 ventanas de tiempo para 6 perfiles de ejecución diferentes.

2) La primera de las configuraciones restantes se obtiene considerando la disipación de potencia media de los distintos componentes en cada uno de los perfiles analizados (AVG). Por lo tanto, el floorplanner busca soluciones factibles que minimicen seis objetivos térmicos (uno por cada perfil) y la longitud del cableado.

3) Se obtiene otra configuración considerando únicamente el mayor consumo de energía por elemento (WOR). Por lo tanto, se atacan tres objetivos: viabilidad, un objetivo térmico y cableado. Este caso corresponde a la estrategia utilizada por otros floorplanners térmicos.

4) Finalmente, se considera una suma ponderada de los consumos de energía de los diferentes perfiles (todos los pesos son iguales) para cada elemento de la arquitectura (WSM). En este caso, el algoritmo trata de optimizar tres objetivos: viabilidad, un objetivo térmico y la longitud del cableado.

De esta manera se propone un enfoque eficiente que incorpora información en forma de perfiles de consumo de potencia para orientar el floorplanning térmico de arquitecturas multiprocesador 3D. El análisis propuesto muestra un compromiso entre el comportamiento térmico y el rendimiento y revela que considerar el consumo de energía en el caso peor no permite encontrar configuraciones óptimas. De hecho,

cuando se trata de minimizar el la suma ponderada del consumo de potencia de los diferentes perfiles de potencia (WSM) se alcanza un mejor comportamiento global. Tal configuración ofrece una mejor respuesta térmica en condiciones extremas reduciendo en 14.01K y 12.84K la temperatura máxima y el gradiente térmico del chip respectivamente.

A.2.4 El impacto del modelo térmico

Como se explica en la Sección 2.5.2, los modelos térmicos utilizados en otros trabajos de floorplanning para evaluar la respuesta térmica de los chips 3D presentan un compromiso entre tiempo de ejecución y precisión. En las propuestas anteriores, el uso de un modelo aproximado está motivado por su bajo coste computacional. Sin embargo, en esta sección, la optimización térmica se guía por primera vez por simulaciones precisas mediante un modelo basado en una red neuronal. De esta manera, se reduce el tiempo de ejecución del algoritmo de floorplanning y se alcanza un mejor comportamiento térmico. Por lo tanto, se obtiene una herramienta óptima para la exploración arquitectónica.

Igual que en las secciones anteriores, empleamos el floorplanner MFA. Por lo tanto, el problema del floorplanning se plantea como un problema multiobjetivo en el que se trata de minimizar la longitud del cableado y la temperatura. Se estudia el impacto de considerar diferentes modelos térmicos, a saber:

1) $APPROX$: El modelo térmico aproximado explicado en la Sección 2.5.2. $APPROX$ es el modelo utilizado en las secciones anteriores.

2) $3D-ICE$: El simulador térmico exacto presentado en la Sección 2.5.2.

3) $NNTM$: El modelo denominado Neural Network Thermal Model introducido en la Sección 2.5.2. Cabe destacar que existen versiones tanto CPU como GPGPU de este modelo.

En esta sección, se demuestra que las soluciones obtenidas con NNTM minimizan tanto la temperatura como la longitud del cableado. La optimización térmica alcanzada se traduce en una reducción del consumo de potencia del 17.4% ya que se

necesita una menor energía para reducir las temperaturas máximas a niveles aceptables. Además, reemplazar el modelo aproximado por una implementación GPU del modelo NNTM conduce a una aceleración de 2.14×. Por lo tanto, NNTM es el modelo más adecuado para la exploración arquitectónica, ya que elimina el compromiso entre precisión y tiempo de ejecución. También se demuestra que son indispensables técnicas de enfriamiento adicionales tales como la refrigeración líquida para mantenerse en el rango de temperaturas viables.

A.2.5 Direct Mapping: una nueva representación para MPSoCs 3D

En la Sección 6.1, se propone una nueva representación que lleva a superar los resultados obtenidos con MFA. Siguiendo la línea de las propuestas explicadas anteriormente, el floorplanning térmico 3D se plantea como un problema de optimización multiobjetivo en el que se trata de minimizar tanto la temperatura como la longitud global del cableado. Al igual que MFA, el floorplanner térmico propuesto se basa en el algoritmo Non Dominated Sorting Genetic Algorithm II (NSGA-II), un conocido Algoritmo Evolutivo Multiobjetivo. Sin embargo, en este caso, se emplea una codificación diferente de la soluciones, a saber, Direct Mapping.

Representación La representación Direct Mapping es idónea para optimizar arquitecturas considerando bloques de alto nivel (núcleos y memorias). Dicha representación es adecuada para problemas floorplanning con área prefijada y permite una traducción directa de individuos del Algoritmo Evolutivo a configuraciones de la arquitectura. Se trata de una ventaja importante ya que el proceso de decodificación de las soluciones no requiere el uso de una heurística encargada de la ubicación de los componentes. Dicha heurística podría limitar el espacio de exploración y causar problemas de convergencia prematura en el caso de los algoritmos evolutivos. Además, con el uso de esta representación, se evita el coste computacional de la etapa de decodificación.

Además, esta representación es compatible con simuladores térmicos actuales tales como 3D-ICE [138] que dividen la superficie del chip en celdas térmicas. De

esta manera, se elimina el error ocasionado por los distintos a los tamaños de celda utilizados en los procesos de optimización y de validación. Por lo tanto, se puede lograr una mejor optimización térmica.

La Figura A.8 muestra una configuración y la correspondiente representación de una arquitectura compuesta de 3 procesadores (C1, C2 y C3) y 3 memorias (L1, L2 y L3). Cada componente de la arquitectura se caracteriza con una coordenada y un valor booleano. La coordenada determina la ubicación de la esquina inferior izquierda del elemento, mientras que el valor booleano indica si el elemento en cuestión ha sido rotado. Esta representación de tipo malla admite un alto nivel de paralelismo y es compatible con el uso de arquitecturas masivamente paralelas como las GPUs. Sin embargo, se deben definir operadores operadores adecuados para asegurar la generación de soluciones factibles y la preservación de la diversidad de la población.

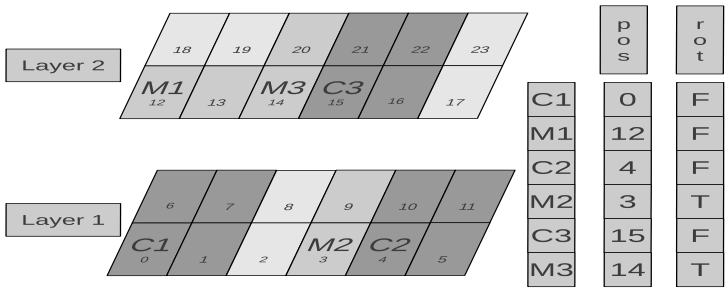


Figure A.8: Configuración de una arquitectura simple y su representación

Población Inicial Se propone un procedimiento rápido y simple para inicializacizar la población. Los componentes se ubican secuencialmente siguiendo un orden aleatorio utilizando una estrategia First Fit (primera posición vacía). Las soluciones obtenidas con este método no son necesariamente configuraciones factibles. Sin embargo, esta heurística evita hasta cierto punto el solapamiento de componentes y las soluciones iniciales presentan un comportamiento aleatorio. Por lo tanto, este método proporciona un buen punto de partida para el proceso de optimización.

Selección El operador de selección implementa una estrategia de torneo binario. Así, se forman parejas aleatoriamente y se selecciona el mejor individuo de cada una para generar la siguiente generación.

Cruce Aplicamos un operador de cruce innovador y específico para este problema. El operador diseñado está basado en una heurística simple que permite obtener soluciones factibles, manteniendo la mayor parte de la información genética de los padres (ver Figura A.9). Para obtener descendientes factibles, se reubica estratégicamente algunos componentes del segundo padre.

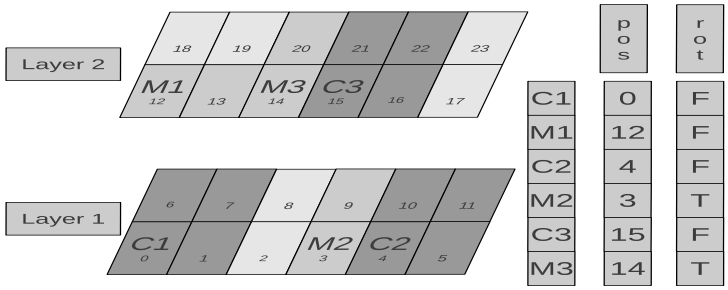
Mutation La mutación de las soluciones se lleva a cabo de tres formas distintas, todas con la misma probabilidad:

1. intercambiando la posición de dos elementos del cromosoma, lo que resulta en un cambio de ubicación de los dos componentes implicados (Figura A.10(b)).

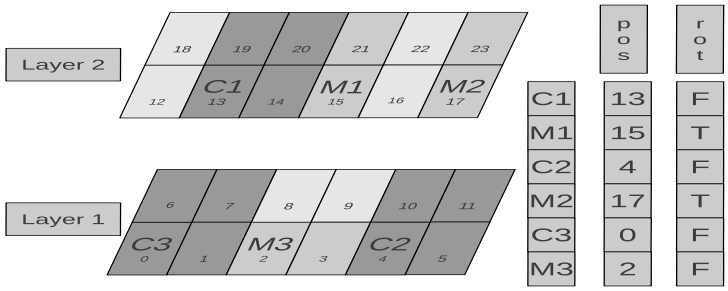
2. rotando un componente (Figura A.10(c)).

3. desplazando aleatoriamente un componente en una de las siguientes direcciones: arriba, abajo, izquierda, derecha, adelante, o atrás (Figura A.10(d)).

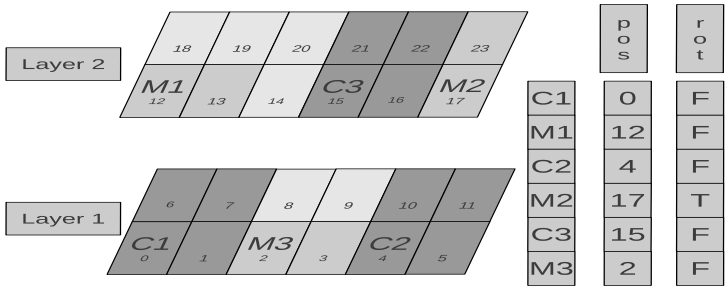
Validación La representación descrita ha sido comparada con propuestas existentes y validadas. Para ello, se amplía el estudio comparativo presentado en el Capítulo 3. Así, se ha demostrado que el uso de la representación Direct Mapping ha llevado a superar los resultados obtenidos con Multiobjective Floorplanning Algorithm, Generalized Polish Expression, Combined Bucket and 2D Array, Double Tree and Sequence y Sequence Pair a la hora de optimizar tres MPSoCs 3D compuestos por 48, 64 y 128 núcleos. El algoritmo y sus operadores correspondientes han sido diseñados con éxito ya que la búsqueda realizada devuelve frentes de configuraciones que abarcan una amplia gama de temperaturas y longitudes de cableado. Además, la representación propuesta no requiere una configuración inicial factible para comenzar la optimización.



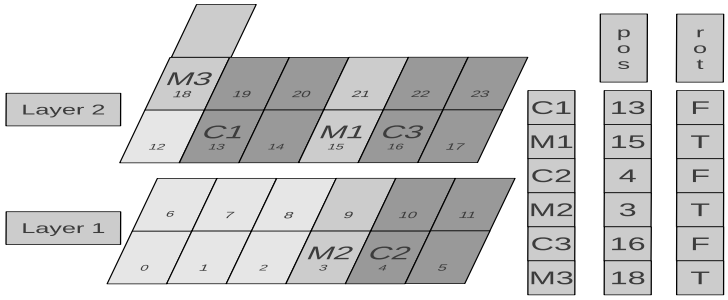
(a) Padre 1



(b) Padre 2

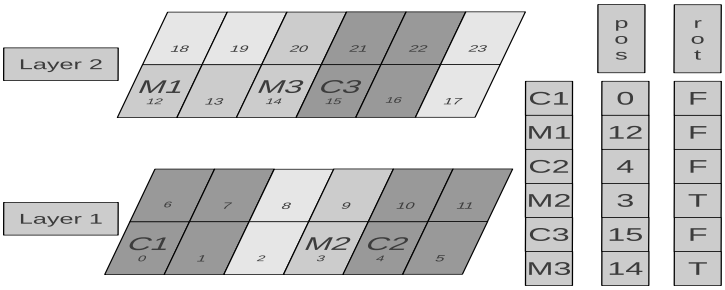


(c) Hijo 1

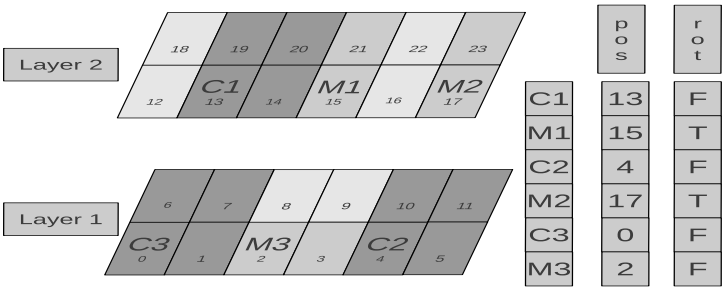


(d) Hijo 2

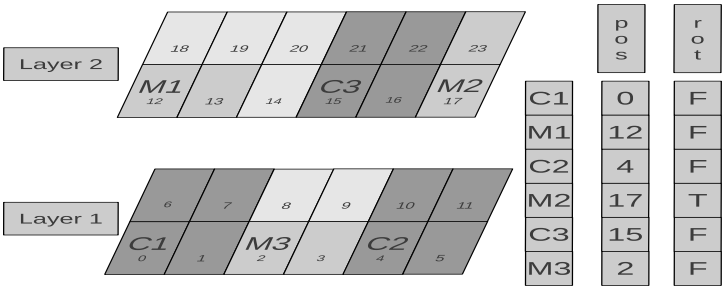
Figure A.9: Operador de cruce: el ejemplo muestra una arquitectura simple compuesta por 6 componentes distribuidos en dos capas.



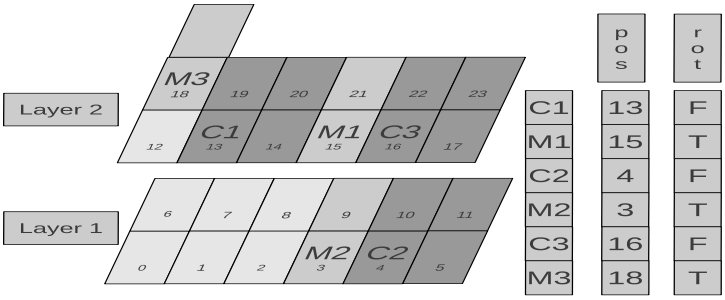
(a) Configuración Original



(b) Intercambio de posición entre M1 y M3



(c) Rotación del componente M1



(d) Desplazamiento del componente M1 (hacia adelante)

Figure A.10: Operador de mutación: el ejemplo muestra una arquitectura simple compuesta por 6 componentes distribuidos en dos capas.

Paralelización Con el fin de proporcionar una herramienta capaz de realizar tareas de exploración de una manera más rápida, se implementa una versión CPU-GPU del floorplanner propuesto. Esta versión proporciona una aceleración de hasta 8.84×. Además, con esta implementación paralela, el tiempo de optimización aumenta linealmente con el tamaño de la arquitectura, garantizando así su escalabilidad.

A.2.6 Optimización con canales de refrigeración líquida

Anteriormente se ha mostrado que las plataformas MPSoCs 3D dotadas de componentes con un alto consumo energético (tales como los núcleos SPARC) alcanzan temperaturas por encima de los límites aceptables. De hecho, son obligatorias técnicas de enfriamiento adicionales cuando se considera este tipo de arquitecturas. Así, se incluyeron canales microfluídicos en las simulaciones térmicas de las configuraciones resultantes. De esta manera, las temperaturas máximas se mantenían en valores factibles. Sin embargo, estos microcanales no se tuvieron en cuenta en el proceso de optimización. Por lo tanto, el floorplanner no podía jugar con las oportunidades de optimización adicionales disponibles en este nuevo escenario. Por ejemplo, la superposición vertical de dos componentes calientes no tiene que ser evitada necesariamente si un canal de refrigeración está situado entre ellos en una capa intermedia del chip.

En la Sección 6.3, se compara la optimización térmica alcanzada con y sin la inclusión de los canales de refrigeración en las simulaciones térmicas realizadas durante el proceso de floorplanning. Con este fin, integramos el simulador 3D-ICE en el floorplanner basado en la representación Direct Mapping. Cabe destacar que la herramienta 3D-ICE permite realizar la simulación térmica de chips 3D con microcanales o sin ellos. Los resultados obtenidos demuestran que la incluir los microcanales en el proceso de optimización lleva a una reducción simultánea de temperatura y cableado en arquitecturas MPSoCs 3D de gran tamaño. En efecto, el floorplanner propuesto es capaz de explotar las nuevas oportunidades de optimización que surgen a partir de la relajación de las restricciones térmicas de las plataformas estudiadas.

A.3 Conclusiones y trabajo futuro

Presentamos aquí las conclusiones de esta tesis. También se discuten posibles mejoras y trabajos futuros que extenderán la investigación presentada.

A.3.1 Conclusiones

En esta tesis, nos hemos enfrentado a un problema relevante y real, el floorplanning térmico de MPSoCS 3D de gran tamaño. La industria de semiconductores se está decantando por chips con un gran número de procesadores. Sin embargo, estas plataformas están limitadas por el elevado número de transacciones de bus y su alta latencia. La integración 3D está considerada como una manera viable de reducir la longitud del cableado global del chip y aumentar el ancho de banda entre las capas conectadas verticalmente por medio de TSVs. Por lo tanto, se espera que estas arquitecturas proporcionen el aumento de rendimiento necesario en los próximos años.

Se ha demostrado la viabilidad de MPSoCs 3D de grandes dimensiones ya que las temperaturas alcanzadas permanecen en niveles aceptables. El floorplanning térmico permite reducir significativamente la temperatura del chip. Sin embargo, las temperaturas máximas prohibitivas que aparecen en las arquitecturas estudiadas hacen necesaria la incorporación de nuevas técnicas de enfriamiento. Se ha simulado la inclusión de canales microfluídicos dispuestos de forma homogénea entre las capas activas del chip 3D. Un refrigerante líquido (agua) fluye a través de estos canales, con un flujo fijo. La técnica considerada permite reducir drásticamente la temperatura del chip, alcanzando temperaturas viables. Se espera que el consumo de energía requerido por dicha técnica de enfriamiento se mantenga en valores admisibles, a pesar de que los valores específicos dependerán del número de canales y de las características específicas de la(s) micro-bomba(s) involucrada(s).

Hemos mostrado la necesidad de nuevas técnicas capaces de lidiar con las limitaciones térmicas impuestas por MPSoCs 3D de grandes dimensiones. Se han adaptado varias propuestas relevantes de floorplanning tales como Combined Bucket and 2D Array, Double Tree and Sequence, Sequence Pair y Generalized Polish Expression para realizar la optimización de tres plataformas basadas en la arquitectura Niagara.

Un estudio comparativo ha mostrado que las técnicas citadas no encuentran soluciones factibles cuando no se proporciona la configuración inicial de la arquitectura. Además, el problema requiere técnicas multiobjetivo ya que los chips estudiados presentan un compromiso entre temperatura y longitud global del cableado. De hecho, las herramientas de floorplanning han de proporcionar una amplia gama de soluciones que presenten un equilibrio adecuado entre los objetivos conflictivos contemplados. De esta manera los fabricantes de estos chips podrán seleccionar la mejor configuración de acuerdo con sus propios criterios.

Se han empleado Algoritmos Evolutivos Multiobjetivo para tratar el compromiso entre rendimiento y temperatura mencionado anteriormente. Estos algoritmos extienden los Algoritmos Evolutivos con la incorporación del concepto de optimalidad de Pareto en el proceso de optimización. Los AEMOs constituyen una herramienta adecuada para hacer frente a problemas multiobjetivo complejos, permitiendo hallar un conjunto de soluciones que optimizan simultáneamente dos o más objetivos conflictivos en un tiempo reducido.

Las nuevas metodologías que se han propuesto permiten la reducción de temperatura y a la mejora de rendimiento de las arquitecturas consideradas. Se ha mejorado de varias maneras una propuesta previa, a saber, el Multiobjective Floorplanning Algorithm (MFA) :

1) En primer lugar, hemos acelerado el proceso de optimización con una implementación paralela. En particular, se ha adoptado un modelo Maestro-Trabajador multi-hilo para obtener una aceleración significativa de la tarea de exploración arquitectónica.

2) Hemos introducido conocimiento dinámico en el proceso floorplanning, que tradicionalmente ha sido tratado como una técnica de optimización estática. Para este fin, se han considerado perfiles de potencia obtenidos mediante simulaciones arquitectónicas. No obstante, incluir estos perfiles en la fase de optimización ha resultado ser una tarea difícil. De hecho, se ha reducido dicha información a una serie de métricas ya que los AEMOs utilizados no son capaces de manejar un número elevado de objetivos.

3) Hemos evaluado el impacto de la integración de diferentes modelos térmicos en el proceso de optimización ya que tradicionalmente se ha asumido un compromiso entre tiempo de ejecución del modelo térmico y su precisión. Hemos considerado el modelo exacto 3D-ICE y un modelo aproximado pero rápido. Además, hemos incorporado en nuestro floorplanner un modelo propuesto recientemente basado en una red neuronal entrenada para reproducir los resultados del simulador 3D-ICE (NNTM). Cabe destacar que este modelo nunca había sido utilizado en un contexto floorplanning. Además, se ha empleado una versión GPGPU de dicho modelo, eliminando el compromiso entre tiempo de ejecución y precisión. De hecho, la red neuronal es más rápida que el modelo térmico aproximado y realiza simulaciones precisas. Por lo tanto, con un tiempo de optimización fijo, las configuraciones alcanzadas con NNTM mejoran los resultados obtenidos con los demás modelos térmicos.

Una contribución importante de esta tesis es la propuesta de Direct Mapping, una nueva representación diseñada para problemas de floorplanning con área fija que permite realizar tareas de exploración arquitectónica de manera eficiente. Direct Mapping explota el potencial del diseño modular habilitado por la integración 3D ya que los distintos componentes de una arquitectura son tratados como bloques IP. Además, es totalmente compatible con herramientas existentes tales como los simuladores térmicos que dividen la superficie de un chip en celdas térmicas rectangulares. Por lo tanto, el método propuesto es integrable en una suite de herramientas que abarquen las diferentes etapas del proceso de diseño de estas arquitecturas, tales como exploración arquitectónica, optimización y validación entre otras. Esta representación permite una asociación directa de los individuos en configuraciones de la arquitectura optimizada, evitando la costosa etapa de decodificación de los algoritmos de optimización. Además, se eliminan heurísticas que pueden limitar la exploración del espacio de búsqueda y causar problemas de convergencia prematura.

Un Algoritmo Evolutivo Multiobjetivo basado en NSGA-II ha sido diseñado para explotar la representación propuesta. La novedosa propuesta de floorplanning ha sido validada y comparada con floorplanners representativos del estado del arte. Los experimentos realizados demuestran que el uso de Direct Mapping junto con oper-

adores específicos para el problema conduce a resultados óptimos, superando todas las técnicas analizadas.

Se ha demostrado que la inclusión de los canales de refrigeración líquida en las capas intermedias de los chips 3D afecta en gran medida al perfil térmico de las arquitecturas estudiadas. Hemos demostrado que es necesario tener en cuenta este efecto de enfriamiento durante el proceso de optimización para conseguir un comportamiento térmico y un rendimiento óptimos. De hecho, los floorplanners deben aprovechar las oportunidades de optimización adicionales proporcionadas por la relajación de las restricciones térmicas.

En resumen, hemos propuesto una serie de heurísticas bioinspiradss que realizan con éxito la optimización de MPSoCs 3D de gran tamaño. Hemos mejorado propuestas anteriores por medio de la introducción de conocimiento específico para el proceso floorplanning en forma de perfiles de potencia, adecuados modelos térmicos y una nueva representación. Además, se han realizado implementaciones paralelas para proporcionar una herramienta eficaz capaz de realizar la exploración arquitectónica de MPSoCs 3D complejos.

A.3.2 Trabajo Futuro

En este trabajo se ha demostrado que las plataformas 3D estudiadas son factibles en términos de temperatura y que proporcionan una mejora de rendimiento debido a la reducción del cableado global. Hay dos formas principales de extender las técnicas presentadas en esta tesis. La primera consiste en considerar nuevas restricciones arquitectónicas en el proceso de optimización. En segundo lugar, se puede mejorar algunas de las técnicas propuestas.

En cuanto a las restricciones arquitectónicas, a pesar de que las tecnologías de integración 3D están en constante evolución, es posible incorporar otras características como una comunicación realista entre procesadores. Por ejemplo, las arquitecturas basadas en la plataforma Niagara estudiadas en este trabajo contienen un conjunto de crossbars para asegurar dicha comunicación. Sin embargo, hoy en día se tiende a implementar una Network-on-Chip (NoC) cuando el número de procesadores in-

tegrados es elevado. En tales diseños, se habilita una comunicación basada en paquetes mediante un procedimiento similar al del Protocolo de Internet. Así, los paquetes pasan a través de switches que se deben ser ubicados apropiadamente. Por otra parte, el número total de estos switches debe ser minimizado para reducir el consumo de energía. El diseño de estas redes integradas representa un problema difícil y está ganando relevancia como tema de investigación [15], [116], [11], y [61]. Del mismo modo que se han empleado simuladores térmicos precisos, es necesario considerar modelos realistas de comunicación. Así, la mejora de rendimiento proporcionada por los MPSoCs 3D estudiados se obtendrá de una manera más precisa.

Otro problema arquitectónico no abordado en este trabajo es la ubicación de las TSVs. Es importante notar que que este problema ya ha sido tratado en el grupo de investigación en trabajos previos de David Cuesta *et al.* [38] y [36], y que estos trabajos son compatibles con las metodologías presentadas en esta tesis. En términos generales, la colocación de TSVs se puede realizar de dos maneras. El primer enfoque consiste en un procedimiento de dos fases en el que se realiza la colocación de TSVs despuś de ubicar los componentes de las arquitecturas o viceversa. La otra posibilidad es considerar que una TSV abarca la totalidad de una celda (“ TSV-as-cell”), de modo que tanto los componentes como las vías son modelados como bloques que deben ser colocados en la misma etapa del proceso de floorplanning. La representación Direct Mapping es compatible con ambas alternativas aunque la segunda podría conllevar un aumento significativo del área total ya que las áreas de los componentes ubicados debe ser al menos igual al tamaño de celda fijado ($300\mu m$ en los experimentos realizados).

La técnica de enfriamiento empleada también es susceptible de ser mejorada como se explica en [39]. Se ha simulado la integración de canales microfluídicos con flujo constante para demostrar la viabilidad de las arquitecturas 3D analizadas. Sin embargo, dependiendo de las características específicas de la bomba utilizada, podría ser posible manejar un flujo variable con el fin de minimizar el consumo de energía, asegurando en todo momento que la temperatura se mantiene en niveles aceptables.

Con respecto a las técnicas propuestas, se han empleado con éxito diferentes AEMOs, alcanzando configuraciones que presentan un buen compromiso entre temperatura y cableado. También se ha demostrado que las configuraciones obtenidas mejoran los resultados obtenidos con otras técnicas validadas de floorplanning. Además, otras propuestas basadas en Simulated Annealing, tales como Combined Bucket and 2D Array no son adecuadas para la exploración arquitectónica de MPSoCs 3D de grandes dimensiones, ya que requieren una configuración inicial factible. De hecho, estas técnicas realizan un ajuste fino de la configuración inicial, principalmente por medio de operadores de búsqueda local. Por tanto, sería beneficioso combinar la capacidad de exploración de los AEMOs estudiados con la explotación llevada a cabo por métodos como CBA. Se podría implementar fácilmente un proceso de optimización dividido en dos etapas en el que, en primer lugar, se obtendría un frente de soluciones no dominadas con un AEMO. Más adelante, se aplicaría el método CBA a las configuraciones más prometedoras del frente obtenido.

También se pueden estudiar otros métodos para aprovechar información dinámica (tales como los perfiles de consumo de potencia) en el proceso de floorplanning. Es importante ver que, en la metodología propuesta en la Sección 5.3, se calculan varias métricas a partir de las trazas de consumo de energía con el fin de guiar la optimización térmica. En un trabajo reciente [157], Wu *et al.* proponen una técnica de Dynamic Voltage and Frequency Scaling (DVFS) que utiliza información dinámica de la carga de trabajo de los distintos componentes de la arquitectura. Las cargas de trabajo también se obtienen mediante simulaciones pero, en este caso, las trazas obtenidas son transformadas al dominio de la frecuencia. Con este elegante enfoque, la cantidad de información manejada por el algoritmo se reduce drásticamente y la información dinámica puede ser explotada por completo.

A.3.3 Discusión

Este trabajo propone el uso de heurísticas bioinspiradas para realizar el floorplanning MPSoCs 3D dotados de un gran número de procesadores. Es importante señalar la diferencia entre los procedimientos de validación de heurísticas y los de validación de arquitecturas, ya que estos dos mundos chocan de alguna manera en algunos aspectos de este trabajo. El funcionamiento adecuado de una heurística debe ser

validado estadísticamente por medio de un proceso experimental reproducible. Por otro lado, en un contexto de diseño arquitectónico, la obtención de una única solución que satisfaga las restricciones existentes, tales como la temperatura, la viabilidad y la longitud de cableado puede ser suficiente. Sin embargo, cuando se trabaja con Algoritmos Evolutivos, no es trivial lidiar con las restricciones mencionadas. De hecho, hemos detectado que la imposición de restricciones duras suele tener un efecto negativo en la dinámica de la búsqueda. Por ejemplo, sería razonable establecer umbrales de temperatura máxima o de longitud de cableado durante el proceso de búsqueda para descartar soluciones candidatas no prometedoras. Sin embargo, hemos observado empíricamente que la imposición de tales restricciones conduce a una exploración parcial del espacio de soluciones, produciendo soluciones subóptimas.

En términos generales, el objetivo es proporcionar las mejores condiciones posibles para el proceso de búsqueda heurística. Tales condiciones óptimas no son intuitivas en la mayoría de los casos. Para ilustrar esta idea, se remite al lector a la Sección 5.3, donde se discute la conveniencia de varias métricas térmicas como función de fitness del proceso de optimización. En dicha sección, se muestra que las temperaturas máximas se alcanzan cuando todos los componentes presentan un consumo de potencia máximo. Sin embargo, no es beneficioso para considerar estos consumos de potencia máxima en el proceso de optimización para reducir los picos de temperatura. De hecho, una relajación de las restricciones térmicas durante el proceso de floorplanning resulta en una mejor optimización térmica.

Otro factor que afecta profundamente la dinámica del proceso de búsqueda es la representación elegida. De hecho, diferentes representaciones implican distintos espacios de soluciones. Estos espacios de soluciones no sólo pueden presentar una cardinalidad diferente, sino también una distribución diferente de los óptimos locales y globales. Por lo tanto, algunas representaciones conllevan Fitness Landscapes más idóneos para búsquedas heurísticas que otras. En la Sección 4.1.4 se proporciona una breve introducción al análisis de Fitness Landscapes en el contexto de la optimización mono-objetivo. Sin embargo, todavía se requieren trabajos de investigación para comprender mejor la información recaudada de dichos análisis. El objetivo fi-

nal del análisis de estos Fitness Landscapes debe ser encontrar algoritmos óptimos. Sin embargo, este proceso se obvia en la mayoría de los trabajos que tratan este tema.

Este tipo de análisis sería muy útil en el contexto de la resolución de problemas complejos en los que normalmente se emplean técnicas heurísticas. El principal reto sigue siendo la gran cardinalidad de los espacios de soluciones estudiadas. Las técnicas de muestreo, tales como Random Walk se emplean generalmente para determinar las propiedades de los Fitness Landscapes analizados (multimodalidad etc.). Algunos trabajos prometedores han revelado que la distribución de óptimos globales y locales en el espacio de soluciones no es aleatoria [42]. Sin embargo, los resultados presentados no son necesariamente generalizables y la técnica propuesta es difícilmente escalable, ya que se basa en procedimientos exhaustivos. El desarrollo actual de técnicas de minería de datos para escenarios BigData podría proporcionar un mayor conocimiento de la distribución de los óptimos buscados. Todas estas dificultades se incrementan cuando se analizan problemas multiobjetivo. En tal caso, es posible analizar la relación entre las soluciones o entre los frentes (conjuntos) de soluciones, como se explica en [151], un prometedor trabajo por S. Verel.

En resumen, se puede afirmar que la investigación realizada no ha hecho más que empezar a descubrir como se relacionan las diferentes soluciones de un problema, como clasificar los distintos problemas en función del análisis de su Fitness Landscape y como diseñar algoritmos eficientes para los problemas analizados.